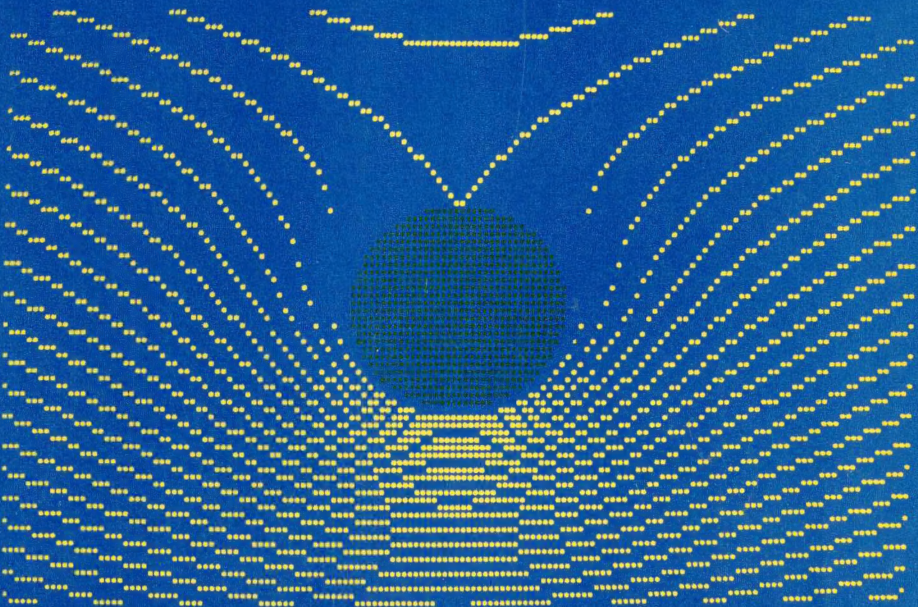
パソコン ライブラリ＝2

各種パソコン用

拡張BASICとその応用

戸川隼人＝著

サイエンス社



各種パソコン用
拡張BASICとその応用
戸川隼人＝著
パソコン ライブラリ＝2
イエンス社

本書の特徴

- 市販されている**ほとんどの機種に利用できる**よう，操作法など初歩からていねいに述べ，機種ごとのBASICの異同も解説されています．

- 学習・記憶・検索等に便利なよう各項が，見やすい**ページ単位で構成**されています．

- 通常のパソコン利用領域における主要事項の学習には**この1冊で間に合う**よう配慮されています．（特に，図形出力，文字列処理，数値計算等を詳説．）

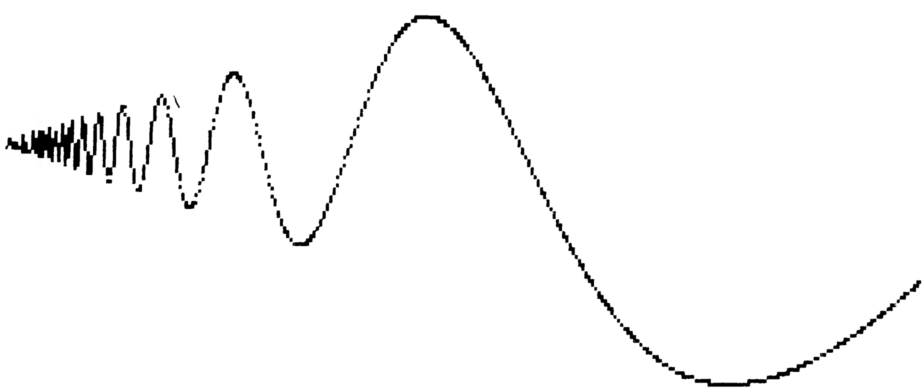
パソコン ライブラリ＝2

各種パソコン用

拡張BASICとその応用

戸川隼人＝著

サイエンス社



まえがき

あなたはどんな乗物が好きですか？　たとえば

　　マイカー　　バス　　新幹線

の中ではどれが好きですか？

私はマイカーが好きです．マイカーは自分の思いどおりに動いてくれて，しかも機動力があるからです．マイカーでなければ行けない，すばらしい観光地がたくさんあります．自家用車をもっていなければできない仕事もたくさんあります．また，クルマそれ自体も非常に魅力的です．数ある工業製品の中で，最もデザインが良く，コスト・パフォーマンスの良いものの一つです．

それと同じ理由で，私はパソコン（＝マイコン）が好きです．パソコンは自分の思いどおりに動いてくれて，しかも，ものすごい潜在的能力をもっています．パソコンを使えば，すばらしいことができます——趣味の上でも，仕事の上でも，勉強の上でも，…….

パソコンを単なる「計算の道具」だと思ってはいけません．パソコンは文字情報を処理する道具です．たとえば，自分で辞書を作れます（本書7章の例題4参照）．1年かけて，個性的な辞書を作ってみませんか？あるいは，統計データーファイルを作るのも意義ある仕事ですし，漢字プリンターを使って，自分で本を作ることもできます．普通は自分で本を作ろうと思っても，自分で活字を組むことはできませんし，印刷屋さんに頼めば高いから，よほどお金がないと自費出版はできません．しかしパソコンを使って版を作り，複写機（またはサーモファックス，簡易オフセットなど）でプリントすれば，自分の本を作れます．

パソコンの文字列処理機能を活用して，プログラムを自動的に書きかえたり，編集したり，解析したりすることができます（本書7章の例題7〜9参照）．自分でオリジナルなプログラミング言語を考え，その処理系を作ることもできます．パソコンのBASICは，そういう処理をするの

に，本当に便利にできています．

パソコンは，また，図形を扱うのに便利な道具です．カラー・グラフィック・ディスプレイ，X–Y プロッター，ライトペン，ディジタイザーなど，図形を扱うための道具がそろっていますし，表示されている図形をメモリーに記憶させたり再表示させたりするためのGET@文，PUT@文など，BASIC の図形処理機能も強力です．

数学や物理で習ったことをパソコンで数値的にやってみたり，グラフを描いてみたりすると，単に式の形で見ているのと違って，具体的によくわかります．本書の表紙はその例で，スピンをかけたボールのまわりの流れです．こういう絵は，パソコンでやれば簡単ですが，手計算ではとても大変で，普通，理くつはわかっていても，実際にはなかなか書けないものです．

じつは，こういう図は，普通のコンピューターで描かせようと思っても，簡単にはできません．あなたの学校(または会社)のコンピューターでできるかどうか調べてごらんなさい．たいていは不可能です．図形入出力の装置が無いからです．なぜ無いかといえば，とても高いからです．また，あったとしても，使い方のわかる人はほんの少ししかいないのです．

パソコンを，単なる「安物のコンピューター」だと思ってはいけません．安いことは事実ですが，同時に，これまでに作られたいかなるコンピューターよりも便利で使い易いのです．最も良いコンピューターが最も安いのですから，この上なく結構なことだと思います．

パソコンは使い易さの点で普通のコンピューターよりもはるかに上等ですが，個人でもてるということが，これまた実にすばらしいことです．昔は，電話も，自動車も，風呂も，「個人でもつのはゼイタク」と考えられておりました．確かにゼイタクには違いありません．個人がもっている電話や自動車や風呂の稼動率はずいぶん低いものです．公衆○○という形でも間にあいます．しかし，やはり，自分でもっているのっていいものですね．皆が一つずつ自分用のをもつのが理想的です．

コンピューターも，そうですよ．パソコンに慣れてしまうと，公衆計

算機などは使う気がしなくなります．冒頭に書いたように，コンピューターを乗物にたとえるならば，

筆算は徒歩
電卓は自転車
プログラム電卓はバイク
パソコンは乗用車
小型コンピューターはバス
中型コンピューターは列車
大型コンピューターは新幹線
スーパー・コンピューターは飛行機

ということになるでしょう．速さの点では，なんといってもスーパー・コンピューターが一番で，また大量の事務処理には中～大型コンピューターが有利ですが，私達のまわりには，そういう処理能力よりも

年中無休．24時間いつでも使える
特定用途に専用に使える
文字処理機能，図形表示機能が強力
手軽にファイル（大容量記憶装置）を使用できる
便利なソフトウェアを容易に入手できる

などの方が重要な仕事がたくさんあります．そういう仕事にはパソコンが一番いいのです．

そういうわけで，将来ますますパソコンが発展することを願って，この本を書きました．全くの初心者から，研究者やマニアまで，いろいろな方々にお楽しみいただけるように書いたつもりです．学校や講習会でも使えるように配慮してあります．演習問題もたくさん付けました．解答は省略しましたが，ほぼ同じ問題の解答が，本書の姉妹編「基本BASICとその応用」に出ていますので，必要なときにはそちらを参照して下さい．

本書は「読むための本」ではありません．パソコンを使って，いろいろやってみるための本です．パソコンは，なるべくご自分でお買いになることをおすすめします．学校の実習の時間にさわるだけ，という程度

では一人前になれません．ピアニスト志望の人で，ピアノをもっていない人はいませんね．１週間に１度，おけいこの日だけ先生の家でピアノにさわる，というだけでは，とてもプロにはなれません．コンピューターも同じです．昔は，自分でコンピューターをもっている人などいませんでしたから，――みんなが素人ばかりでしたから，それでも一人前ということになっていましたが，これからはそうは行きません．現在，すでにかなりの学生がパソコンをもっており，程度の高いプログラムを書いています．また，小学校にもパソコン・クラブを作れそうなくらい普及してきております．そういう子供達に負けないように，ひとつ，がんばって勉強して下さい．

1982年６月

戸川隼人

目　　次

5　基本的なプログラムの技法

6　数 値 計 算

[illegible]

1　機 器 構 成

1.1　CPU

パーソナル・コンピューター（personal computer）のことを略してパソコンという．

personal は「個人の」という意味である．パソコンの **CPU**（central processing unit，**中央処理装置**）は，こんなに小さい（下の写真は実物大）．



Z-80A

NEC の PC-8000 シリーズや PC-6001
シャープの MZ-80 シリーズ，東芝の PASOPIA
SORD の 200 シリーズ，SFC の BUBCOM80
沖電気の *if*-800 などに使われている．

メモリー（記憶装置）も，こんなに小さい．



これで 64K ビット，すなわち
8K バイトの容量がある．

1.2 本 体

CPU は単独では動かない．パソコンとして使うには，そのほかに

電 源

メモリー(記憶装置)

インターフェース(外部の機器を接続するための窓口)

キーボード(鍵盤)

などが，最低限，必要である．これらをまとめてケースにおさめたものをパソコンの**本体**という．本体は，こんな形をしている．



PC-8801



Micoro-8



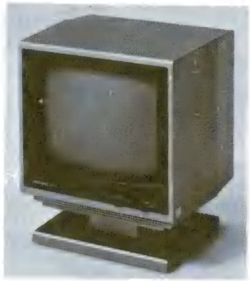
MB-6890 (通称 Level-3)



PC-6001

1.3 ディスプレイ

パソコンの操作をしたり計算結果を見たりするには，**ディスプレイ装置**（**モニター**ともいう）が必要である．最低限，本体とディスプレイ装置があれば，パソコンとして使うことができる．



PASOPIA のディスプレイ

本体とディスプレイ装置が一体になっている機種もある．



MZ-80B



***if*-800**

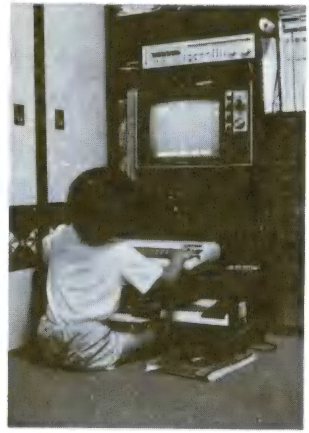


Multi-16



N5200-05

家庭用テレビをディスプレイ装置の代用品として使うこともできる.

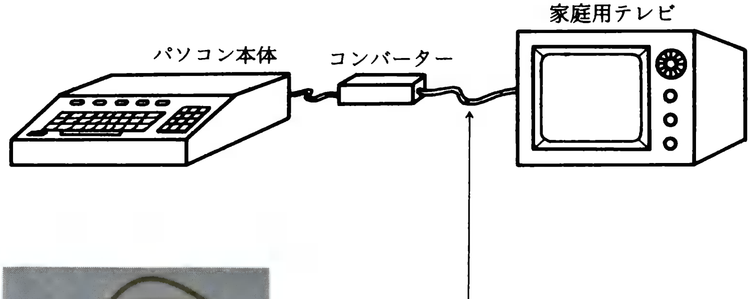


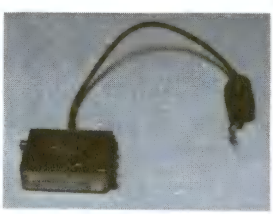
[備考]

1）接続の要領はテレビ・ゲームと同様で，チャンネル2または1（空いている方）で見ることができる.

2）家庭用テレビに接続するには，パソコンの出力をテレビと同じ高周波信号に変換する必要がある．そのための変換器（アダプター，コンバーター）を内蔵している機種もあるが，内蔵していない機種も多いので，購入時に確認すること.

3）家庭用テレビでも，図形は比較的きれいに表示され，色も悪くない．しかし，1行に表示できる字数や，1画面に表示できる行数が少ないので，プログラムを開発したり本格的な計算をしたりするにはやはり専用のディスプレイ装置を使う方がよい.





ここにアンテナとの切換えスイッチを付けておくと便利である．筆者は，任天堂のテレビ・ゲーム（レーシング112）の付属品を流用している（写真左）．なお，切換えスイッチ付きのコンバーターも市販されている.

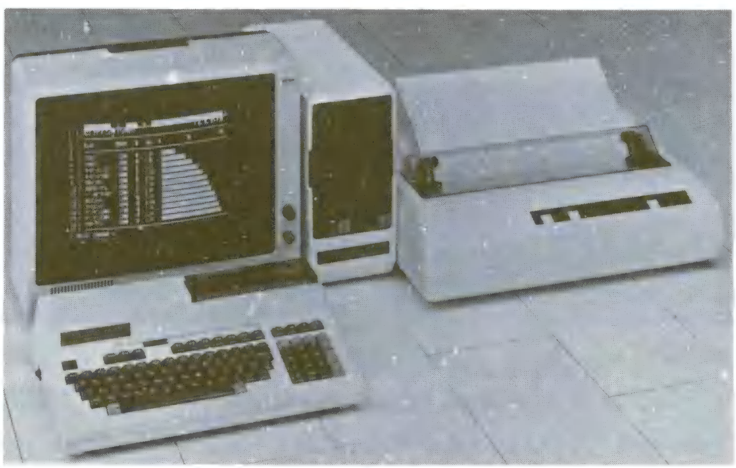
[パソコン・システムの例]



PC-8001



PASOPIA



SORD M-23

1.4　カセット・テープ

目　的　パソコンのメモリーの内容は，電源を切ると消えてしまう．入力したデーターやプログラムを後日，再び使用したければ，再入力可能な媒体に記録しておく必要がある．そのような目的にパソコンで利用できるものとしては

　　カセット・テープ
　　フロッピー・ディスク
　　その他（バブル・カセット，不揮発性RAMなど）

があるが，その内で最も安く手軽に利用できるのがカセット・テープである．カセット・テープはまた，ソフトウェアの流通媒体としても広く用いられている．

テープ　テープは普通のオーディオ・カセット・テープでよい．高級品は必要なく，1本200円前後の安いテープで十分まにあう．音楽的特性よりも，テープに折れ曲がりのないことの方が重要である．長時間テープは折れ曲がりやすく，頭出しや巻きもどしに時間がかかるなど不便なことが多いので，パソコンではもっぱら，C-10（片面で5分，両面で10分．以下同様），C-15，C-20などが使われる．かなり長いプログラムでも，C-10の片面に入ってしまう．



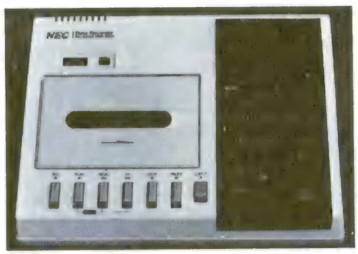


シャープのC-15　　マクセルのC-20

パソコン用カセット・テープ

テープ・レコーダー　テープ・レコーダーも，原理的には，要するに録音と再生ができればよいわけで，それ以上の特殊な機能は要求されていない．もしも手持ちのテープ・レコーダーがあれば，まずそれを使ってみるとよい．ただし，一般のテープ・レコーダーを使用した場合，録音レベルと再生レベルの調節はきわめてデリケートで，調子よく動くようにするためには，何十回でも，何百回でも根気よく試行錯誤をくりかえす必要がある．

それに対し，パソコン専用に開発されたレコーダー PC-6082（日本電気製であるが各社のパソコンに使用可能）は，かなり広いレンジにわたって確実に動作してくれる．その上，前後五つまでのプログラムの自動頭出しができて便利である．



PC-6082

シャープの MZ-80 シリーズは，本体にカセット・レコーダーが組み込まれている．本機のカセット・レコーダーも性能がよく，安心して使用できる．



MZ-80K2

互換性　異なる機種の間のカセット・テープの互換性は無い．すなわち，あるパソコンで記憶したテープを他社のパソコンで読み取ることは，通常の方法では不可能である．

1.5　フロッピー・ディスク

ディスクは円盤の表面に磁気的に情報を記録するもので，

大量のデーターを記録できる

ランダム・アクセスが可能*

そのわりに安い

などの特長があり，大容量記憶装置として，また情報を保管する手段として，広く用いられている．

フロッピー・ディスクは，それを小型コンピューターのために簡易化したもので，円盤は厚紙のジャケットに入っており，そのまま駆動装置にさし込んで使用し，使用後はとりはずして保管することができる．

標準フロッピー

ミニフロッピー



5インチ
(約 13 cm)

8インチ
(約 20 cm)

小型コンピューターで普通，使われているのは，標準フロッピー(8インチ・フロッピー)と呼ばれているもの（写真右）で，それを使用できるパソコンもあるが，かなり高価なので，パソコン用に，ひとまわり小さいミニフロッピー（5インチ・フロッピー，写真左）が開発され，パソコンでは主にこちらが使われている．なお，さらに小さなマイクロ・フロッピーも開発され，一部の機種で使用されている．

---

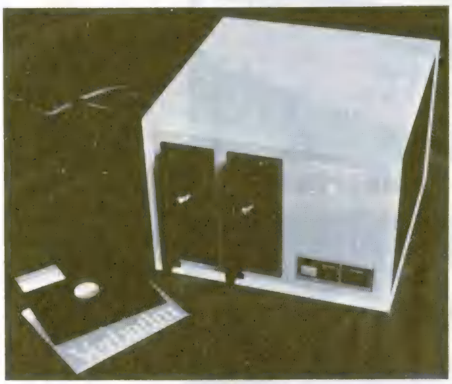
*　どんな順序でも即座にデーターの書き込み，読み出しができること．

容　量　　フロッピー・ディスクの記録方式には

片面だけに記録する	単密度
両面に記録する	倍密度

の区別があり，それによって容量は（機種によって多少の差はあるが）だいたい，次のようになる*.

		ミニフロッピー	標準フロッピー
片面	単密度	75KB	250KB
	倍密度	150	500
両面	単密度	150	500
	倍密度	300	1MB



ミニフロッピーの駆動装置の例（日立 MP3540）



標準フロッピーの駆動装置の例（日本電気 PC-8881）

インターフェース　　ディスク装置を本体に直結できる機種もあるが，接続のための仲介装置（インターフェース）を必要とする機種も多い.

---

* KB はキロバイト（≒ 1000 バイト = 250 ワード）．1 バイト（byte）は 8 ビットで，文字や記号 1 字に対応．たとえば容量が 100KB ならば十万字記憶できることになる．MB はメガバイト（≒ 百万バイト）.

1.6 プリンター

パソコンでは主にドット・プリンターが使われている．これは活字を使わず，文字の形に点を打ってゆく方式のプリンターで，

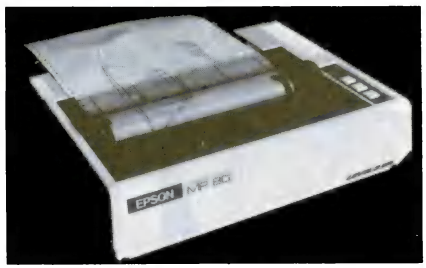
安く，小型，軽量で速い

文字の種類を多くできる

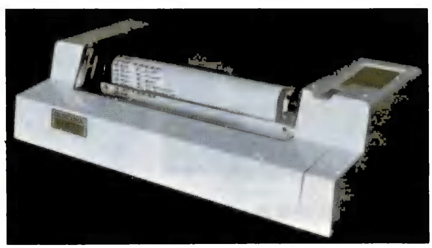
文字の大きさを変えることも容易

図形出力も可能

などの利点がある．現在，下記の2機種が広く使われている．



EPSON（信州精器）の MP-80



精工舎の GP-80

GP-80の方が小型で紙幅も狭いが文字の間隔も狭いので，両機とも1行に80字（縮小文字を使えば132字）打てる．なお，

横幅がもっと広いもの

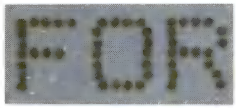
漢字が使用できるもの（縦書き可能の機種もある）

赤と黒の切り換えのできるもの

活字を使って本格的な美しいフォントで印字できるもの

なども市販されている．

ドット・プリンターによる印字の拡大写真

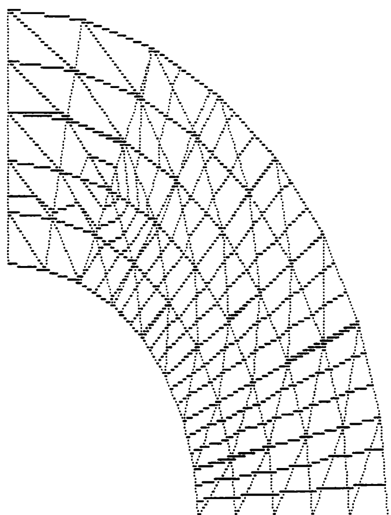


FOR

ドット・プリンターによる出力例

```
  ! " # $ % & ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; < = > ?
@ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [ ¥ ] ^ _
` a b c d e f g h i j k l m n o p q r s t u v w x y z { ¦ } ~ \
年 月 日 市 区 町 を ぁ ぃ ぅ ぇ ぉ ゃ ゅ ょ っ π あ い う え お か き く け こ さ し す せ そ
テ ■ 「 」 、 ・ ヲ ァ ィ ゥ ェ ォ ャ ュ ョ ッ ー ア イ ウ エ オ カ キ ク ケ コ サ シ ス セ ソ
タ チ ツ テ ト ナ ニ ヌ ネ ノ ハ ヒ フ ヘ ホ マ ミ ム メ モ ヤ ユ ヨ ラ リ ル レ ロ ワ ン ゛ ゜
た ち つ て と な に ぬ ね の は ひ ふ へ ほ ま み む め も や ゆ よ ら り る れ ろ わ ん ▒
```

グラフや図形も描ける



漢字の出力例

新年おめでとうございます。

＊ 最近、マイコンにこっています。隼人

＊ 40の手習いでヴァイオリンのおけいこを初めました。保子

＊ 演劇部に入りました。久詩

＊ ハムスターとカメをかっています。 竹詩

＊ つりの本を毎日読んでいます。 清詩

1.7 そ の 他

以上が標準的な構成機器であるが，そのほか必要に応じて各種の周辺装置を接続して利用することができる．

ライトペン　ディスプレイ画面をペンで指して，項目の選択や座標の入力を行なうことができる．

ジョイスティック　写真（右）のような棒状のスイッチで，棒を倒している向き（8方向）を検出できる．

X–Y プロッター　X 方向，Y 方向に自由にペンを動かして図形や文字を描かせることができる．何本かのペンを自動的に変換して多色のプロットのできる機種もある．

ディジタイザー　図面を盤上に固定し，ペンで図面上の点を指して座標を読み取ることができる．



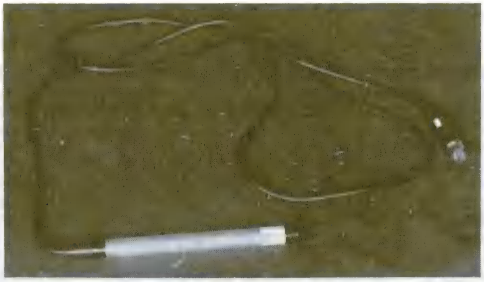
音響カプラー　電話で計算センター（またはパソコンを持っている友人）を呼び出し，受話器を音響カプラーにはめると，データーやプログラムを送信受信できる．

AD (analog to digital) コンバーター　電圧を数値データーに変換して入力する装置．自動計測などに用いられる．

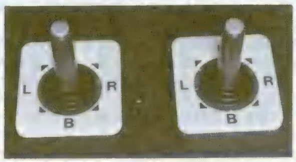
DA (digital to analog) コンバーター　コンピューターの出力を電圧に変換する装置．自動制御などに用いられる．

接点入力装置　外部のスイッチ（あるいはリレー接点など）の ON，OFF を検出して入力する装置．

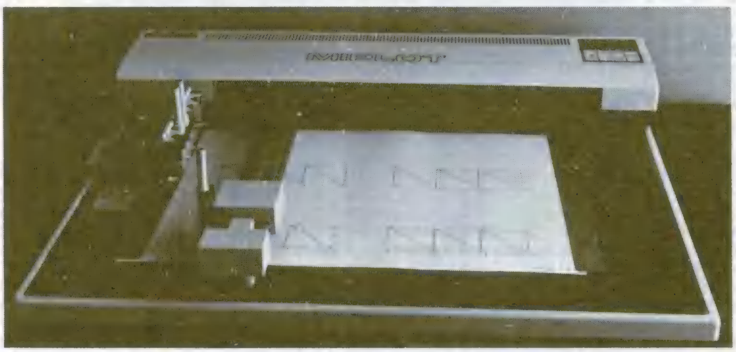
接点出力装置　コンピューターの指令によってスイッチを ON，OFF する装置．



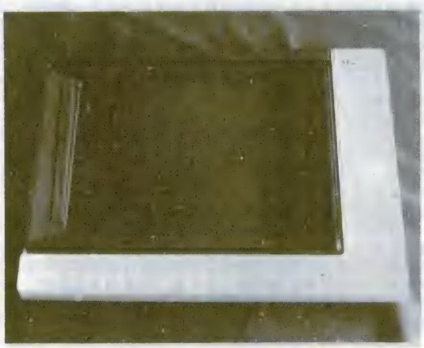
ライトペン



ジョイスティック



X-Y プロッター



簡易ディジタイザー

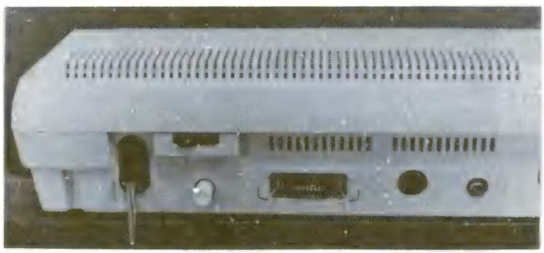
[資料] パソコンで利用できる漢字（PC-8801）

亜唖娃阿哀愛挨姶逢葵茜穐悪握渥旭葦芦鯵梓圧斡扱宛姐虻飴絢綾鮎或粟袷安庵按暗案闇鞍杏以伊位依偉囲
夷委威尉惟意慰易椅為畏異移維緯胃萎衣謂違遺医井亥域育郁磯一壱溢逸稲茨芋鰯允印咽員因姻引飲淫胤蔭
院陰隠韻吋右宇烏羽迂雨卯鵜窺丑碓臼渦嘘唄欝蔚鰻姥厩浦瓜閏噂云運雲荏餌叡営嬰影映曳栄永泳洩瑛盈穎
頴英衛詠鋭液疫益駅悦謁越閲榎厭円園堰奄宴延怨掩援沿演炎焔煙燕猿縁艶苑薗遠鉛鴛塩於汚甥凹央奥往応
押旺横欧殴王翁襖鴬鴎黄岡沖荻億屋憶臆桶牡乙俺卸恩温穏音下化仮何伽価佳加可嘉夏嫁家寡科暇果架歌河
火珂禍禾稼箇花苛茄荷華菓蝦課嘩貨迦過霞蚊俄峨我牙画臥芽蛾賀雅餓駕介会解回塊壊廻快怪悔恢懐戒拐改
魁晦械海灰界皆絵芥蟹開階貝凱劾外咳害崖慨概涯碍蓋街該鎧骸浬馨蛙垣柿蛎鈎劃嚇各廓拡撹格核殻獲確穫
覚角赫較郭閣隔革学岳楽額顎掛笠樫橿梶鰍潟割喝恰括活渇滑葛褐轄且鰹叶椛樺鞄株兜竃蒲釜鎌噛鴨栢茅萱
粥刈苅瓦乾侃冠寒刊勘勧巻喚堪姦完官寛干幹患感慣憾換敢柑桓棺款歓汗漢澗潅環甘監看竿管簡緩缶翰肝艦
莞観諌貫還鑑間閑関陥韓館舘丸含岸巌玩癌眼岩翫贋雁頑顔願企伎危喜器基奇嬉寄岐希幾忌揮机旗既期棋棄
機帰毅気汽畿祈季稀紀徽規記貴起軌輝飢騎鬼亀偽儀妓宜戯技擬欺犠疑祇義蟻誼議掬菊鞠吉吃喫桔橘詰砧杵
黍却客脚虐逆丘久仇休及吸宮弓急救朽求汲泣灸球究窮笈級糾給旧牛去居巨拒拠挙渠虚許距鋸漁禦魚亨享京
供侠僑兇競共凶協匡卿叫喬境峡強彊怯恐恭挟教橋況狂狭矯胸脅興蕎郷鏡響饗驚仰凝尭暁業局曲極玉桐粁僅
勤均巾錦斤欣欽琴禁禽筋緊芹菌衿襟謹近金吟銀九倶句区狗玖矩苦躯駆駈駒具愚虞喰空偶寓遇隅串櫛釧屑屈
掘窟沓靴轡窪熊隈粂栗繰桑鍬勲君薫訓群軍郡卦袈祁係傾刑兄啓圭珪型契形径恵慶慧憩掲携敬景桂渓畦稽系
経継繋罫茎荊蛍計詣警軽頚鶏芸迎鯨劇戟撃激隙桁傑欠決潔穴結血訣月件倹倦健兼券剣喧圏堅嫌建憲懸拳捲
検権牽犬献研硯絹県肩見謙賢軒遣鍵険顕験鹸元原厳幻弦減源玄現絃舷言諺限乎個古呼固姑孤己庫弧戸故枯
湖狐糊袴股胡菰虎誇跨鈷雇顧鼓五互伍午呉吾娯後御悟梧檎瑚碁語誤護醐乞鯉交佼侯候倖光公功効勾厚口向
后喉坑垢好孔孝宏工巧巷幸広庚康弘恒慌抗拘控攻昂晃更杭校梗構江洪浩港溝甲皇硬稿糠紅紘絞綱耕考肯肱
腔膏航荒行衡講貢購郊酵鉱砿鋼閤降項香高鴻剛劫号合壕拷濠豪轟麹克刻告国穀酷鵠黒獄漉腰甑忽惚骨狛込
此頃今困坤墾婚恨懇昏昆根梱混痕紺艮魂些佐叉唆嵯左差査沙瑳砂詐鎖裟坐座挫債催再最哉塞妻宰彩才採栽
歳済災采犀砕砦祭斎細菜裁載際剤在材罪財冴坂阪堺榊肴咲崎埼碕鷺作削咋搾昨朔柵窄策索錯桜鮭笹匙冊刷
察拶撮擦札殺薩雑皐鯖捌錆鮫皿晒三傘参山惨撒散桟燦珊産算纂蚕讃賛酸餐斬暫残仕仔伺使刺司史嗣四士始
姉姿子屍市師志思指支孜斯施旨枝止死氏獅祉私糸紙紫肢脂至視詞詩試誌諮資賜雌飼歯事似侍児字寺慈持時
次滋治爾璽痔磁示而耳自蒔辞汐鹿式識鴫竺軸宍雫七叱執失嫉室悉湿漆疾質実蔀篠偲柴芝屡蕊縞舎写射捨赦
斜煮社紗者謝車遮蛇邪借勺尺杓灼爵酌釈錫若寂弱惹主取守手朱殊狩珠種腫趣酒首儒受呪寿授樹綬需囚収周
宗就州修愁拾洲秀秋終繍習臭舟蒐衆襲讐蹴輯週酋酬集醜什住充十従戎柔汁渋獣縦重銃叔夙宿淑祝縮粛塾熟
出術述俊峻春瞬竣舜駿准循旬楯殉淳準潤盾純巡遵醇順処初所暑曙渚庶緒署書薯藷諸助叙女序徐恕鋤除傷償
勝匠升召哨商唱嘗奨妾娼宵将小少尚庄床廠彰承抄招掌捷昇昌昭晶松梢樟樵沼消渉湘焼焦照症省硝礁祥称章
笑粧紹肖菖蒋蕉衝裳訟証詔詳象賞醤鉦鍾鐘障鞘上丈丞乗冗剰城場壌嬢常情擾条杖浄状畳穣蒸譲醸錠嘱埴飾
拭植殖燭織職色触食蝕辱尻伸信侵唇娠寝審心慎振新晋森榛浸深申疹真神秦紳臣芯薪親診身辛進針震人仁刃
塵壬尋甚尽腎訊迅陣靭笥諏須酢図厨逗吹垂帥推水炊睡粋翠衰遂酔錐錘随瑞髄崇嵩数枢趨雛据杉椙菅頗雀裾
澄摺寸世瀬畝是凄制勢姓征性成政整星晴棲栖正清牲生盛精聖声製西誠誓請逝醒青静斉税脆隻席惜戚斥昔析
石積籍績脊責赤跡蹟碩切拙接摂折設窃節説雪絶舌蝉仙先千占宣専尖川戦扇撰栓栴泉浅洗染潜煎煽旋穿箭線
繊羨腺舛船薦詮賎践選遷銭銑閃鮮前善漸然全禅繕膳糎噌塑岨措曾曽楚狙疏疎礎祖租粗素組蘇訴阻遡鼠僧創
双叢倉喪壮奏爽宋層匝惣想捜掃挿掻操早曹巣槍槽漕燥争痩相窓糟総綜聡草荘葬蒼藻装走送遭鎗霜騒像増憎
臓蔵贈造促側則即息捉束測足速俗属賊族続卒袖其揃存孫尊損村遜他多太汰詑唾堕妥惰打柁舵楕陀駄騨体堆
対耐岱帯待怠態戴替泰滞胎腿苔袋貸退逮隊黛鯛代台大第醍題鷹滝瀧卓啄宅托択拓沢濯琢託鐸濁諾茸凧蛸只
叩但達辰奪脱巽竪辿棚谷狸鱈樽誰丹単嘆坦担探旦歎淡湛炭短端箪綻耽胆蛋誕鍛団壇弾断暖檀段男談値知地
弛恥智池痴稚置致蜘遅馳築畜竹筑蓄逐秩窒茶嫡着中仲宙忠抽昼柱注虫衷註酎鋳駐樗瀦猪苧著貯丁兆凋喋寵
帖帳庁弔張彫徴懲挑暢朝潮牒町眺聴脹腸蝶調諜超跳銚長頂鳥勅捗直朕沈珍賃鎮陳津墜椎槌追鎚痛通塚栂掴
槻佃漬柘辻蔦綴鍔椿潰坪壷嬬紬爪吊釣鶴亭低停偵剃貞呈堤定帝底庭廷弟悌抵挺提梯汀碇禎程締艇訂諦蹄逓
邸鄭釘鼎泥摘擢敵滴的笛適鏑溺哲徹撤轍迭鉄典填天展店添纏甜貼転顛点伝殿澱田電兎吐堵塗妬屠徒斗杜渡
登菟賭途都鍍砥砺努度土奴怒倒党冬凍刀唐塔塘套宕島嶋悼投搭東桃梼棟盗淘湯涛灯燈当痘祷等答筒糖統到
董蕩藤討謄豆踏逃透鐙陶頭騰闘働動同堂導憧撞洞瞳童胴萄道銅峠鴇匿得徳涜特督禿篤毒独読栃橡凸突椴届
鳶苫寅酉瀞噸屯惇敦沌豚遁頓呑曇鈍奈那内乍凪薙謎灘捺鍋楢馴縄畷南楠軟難汝二尼弐迩匂賑肉虹廿日乳入
如尿韮任妊忍認濡禰祢寧葱猫熱年念捻撚燃粘乃廼之埜嚢悩濃納能脳膿農覗蚤巴把播覇杷波派琶破婆罵芭馬
俳廃拝排敗杯盃牌背肺輩配倍培媒梅楳煤狽買売賠陪這蝿秤矧萩伯剥博拍柏泊白箔粕舶薄迫曝漠爆縛莫駁麦
函箱硲箸肇筈櫨幡肌畑畠八鉢溌発醗髪伐罰抜筏閥鳩噺塙蛤隼伴判半反叛帆搬斑板氾汎版犯班畔繁般藩販範
釆煩頒飯挽晩番盤磐蕃蛮匪卑否妃庇彼悲扉批披斐比泌疲皮碑秘緋罷肥被誹費避非飛樋簸備尾微枇毘琵眉美
鼻柊稗匹疋髭彦膝菱肘弼必畢筆逼桧姫媛紐百謬俵彪標氷漂瓢票表評豹廟描病秒苗錨鋲蒜蛭鰭品彬斌浜瀕貧
賓頻敏瓶不付埠夫婦富冨布府怖扶敷斧普浮父符腐膚芙譜負賦赴阜附侮撫武舞葡蕪部封楓風葺蕗伏副復幅服
福腹複覆淵弗払沸仏物鮒分吻噴墳憤扮焚奮粉糞紛雰文聞丙併兵塀幣平弊柄並蔽閉陛米頁僻壁癖碧別瞥蔑箆
偏変片篇編辺返遍便勉娩弁鞭保舗鋪圃捕歩甫補輔穂募墓慕戊暮母簿菩倣俸包呆報奉宝峰峯崩庖抱捧放方朋
法泡烹砲縫胞芳萌蓬蜂褒訪豊邦鋒飽鳳鵬乏亡傍剖坊妨帽忘忙房暴望某棒冒紡肪膨謀貌貿鉾防吠頬北僕卜墨
撲朴牧睦穆釦勃没殆堀幌奔本翻凡盆摩磨魔麻埋妹昧枚毎哩槙幕膜枕鮪柾鱒桝亦俣又抹末沫迄侭繭麿万慢満
漫蔓味未魅巳箕岬密蜜湊蓑稔脈妙粍民眠務夢無牟矛霧鵡椋婿娘冥名命明盟迷銘鳴姪牝滅免棉綿緬面麺摸模
茂妄孟毛猛盲網耗蒙儲木黙目杢勿餅尤戻籾貰問悶紋門匁也冶夜爺耶野弥矢厄役約薬訳躍靖柳薮鑓愉愈油癒
諭輸唯佑優勇友宥幽悠憂揖有柚湧涌猶猷由祐裕誘遊邑郵雄融夕予余与誉輿預傭幼妖容庸揚揺擁曜楊様洋溶
熔用窯羊耀葉蓉要謡踊遥陽養慾抑欲沃浴翌翼淀羅螺裸来莱頼雷洛絡落酪乱卵嵐欄濫藍蘭覧利吏履李梨理璃
痢裏裡里離陸律率立葎掠略劉流溜琉留硫粒隆竜龍侶慮旅虜了亮僚両凌寮料梁涼猟療瞭稜糧良諒遼量陵領力
緑倫厘林淋燐琳臨輪隣鱗麟瑠塁涙累類令伶例冷励嶺怜玲礼苓鈴隷零霊麗齢暦歴列劣烈裂廉恋憐漣煉簾練聯
蓮連錬呂魯櫓炉賂路露労婁廊弄朗楼榔浪漏牢狼篭老聾蝋郎六麓禄肋録論倭和話歪賄脇惑枠鷲亙亘鰐詫藁蕨
椀湾碗腕

2 操　作　法

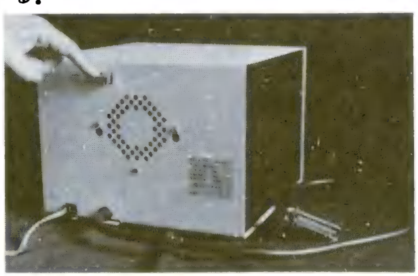
2.1 電　源　ON

本体の電源スイッチは，たいてい，後側（またはフタの中）についている．これは使用中に誤って（あるいはイタズラなどのために）電源を切ることがないようにするためである．使用中に電源を切ると，メモリーの内容は全部失われ，使用中のディスク・ファイルも被害を受けることがある．



PC-6001 の本体を後から見たところ

フロッピー・ディスクの電源スイッチも，使用中に ON，OFF するとディスクに記録された内容が破壊されるので，誤操作防止のためスイッチは後についている．

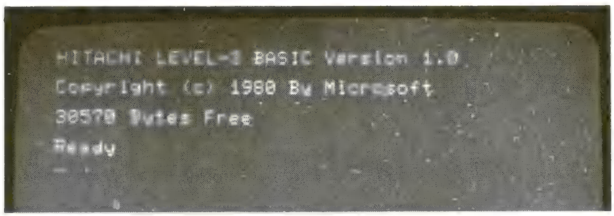


フロッピー・ディスク装置 MP-3540 を後から見たところ

その他の機器は，かなり自由に ON，OFF できる．

2.2 BASIC の起動

多くの機種，たとえば日立の Level-3，富士通の Micro-8，シャープの MZ-80B(ディスク・ベースで使用する場合)，日本電気の PC-8001(ディスクを使用しない場合)，などにおいては，電源を入れるとただちに BASIC を使用できる状態になる．



電源投入直後の画面(日立の Level-3 の例)
Ready とか OK とかいう表示は，BASIC の入力を受け付けられる状態にあることを表す．

初期設定のための簡単な応答を必要とする機種もある．以下はその二，三の例である．

PC-6001* 電源を入れると

　　How many pages ?

というメッセージが出るから，

　　グラフィックを使わない（文字表示だけでよい）場合は 1 [改行]

　　グラフィックを使う場合は（普通）2 [改行]

　　複数箇のグラフィック画面を使いたい場合は 3 または 4 [改行]

を入力する．

PC-8001*（ディスク使用の場合）　　電源を入れると

　　How many files ?

というメッセージが出るから，同時にオープンするファイルの数を入力する（普通，2 を入れておけば安全）．

---

* いずれも日本電気の製品．

左記の諸機種でBASICがすぐ使えるのは，BASICのインタープリーター（プログラムを解読して実行する仕掛け）が本体に組み込まれているからである．それに対し，BASICのインタープリーターを内蔵していないパソコンもある．そういう機種でBASICを使う際には，まずBASICインタープリーターをカセット・テープなどから読み込む必要がある*．たとえば；

MZ-80** 電源を入れると，まず，モニター（実行管理プログラム）の直接制御下に入り，モニターへのコマンド（指令）を入力できる状態になる（そういう状態のときは*印が出る）．そこでBASICインタープリーターのテープをセットし，キーボードから

LŌAD [改行]

を入れると読み込みが始まる．1分40秒後，テープが止まってREADYが表示されたらBASICを使用できる．



BASICのシステム・テープ

BUBCOM80*** BASICインタープリーターの入っているバブル・カセットまたはフロッピー・ディスクを装着して，リセット・ボタンを押す．

---

* これはBASICの利用者にとっては不便であるが，他機種でもBASIC以外のプログラミング言語(FORTRAN, COBOL, PASCALなど)を使う場合には同様の操作が必要である．

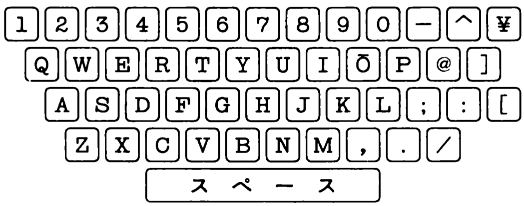
** シャープのMZ-80B, MZ-80C, MZ-80K2などをテープ・ベース(ディスクなし）で使用する場合

*** システムズ・フォーミュレート社（SFC）の製品

2.3 キーボード

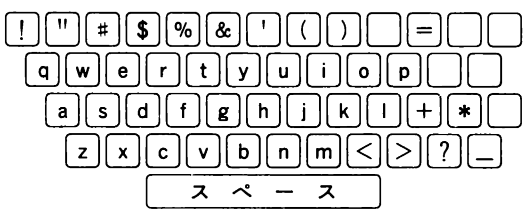
キーボート（鍵盤）におけるキーの配置に関してはJISの規定があり，だいたいどの機種もこれに従っている．

普通の（シフトしない）状態では次のようになっている*．



これはタイプライターと同じ配列である．ただし，多くのパソコンでは，普通の状態では大文字が入力され，シフトしたときに小文字が入力される（PC-8001，PC-6001，BUBCOM80などのように，標準状態が小文字，シフトすると大文字，という機種もあるが，BASICのプログラムの入力時には小文字で入力しても，注釈と文字列定数以外はすべて大文字に変換される）．

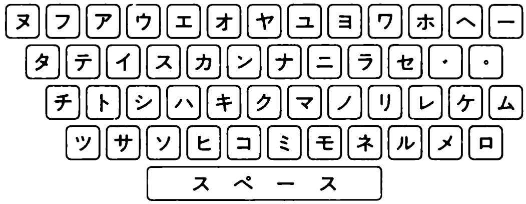
上段シフトした状態では次のようになる．



（PC-8001，PC-6001，BUBCOM80などの場合は，こちらが大文字になる）

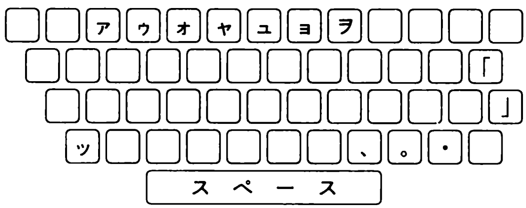
＊ 状態の切換えは，上段下段シフト・キー，英数字とカナ文字の切換えキー，グラフィック切換えキーによって行なう．

カナ文字シフトの状態においては次のようになる．

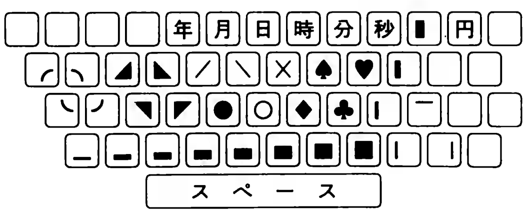


これは非常に覚えにくい配列で，最初のうちはカルタ取りのような感じになるが，打ちにくいからといって使わなければ，いつまでも上達しないから，タイプライターの練習だと思って根気よくまめに使うことが大切である．

カナ文字の上段は次のようになっている．



グラフィック・シンボルのキー配列は機種によって異なる．その一例として，PC-8001，PC-8801 の配列を以下に示す．



2.4 プログラムの入力

基本的な入力方法　BASICのプログラムやデーターは，行単位に入力する規則になっている．それらの行の終りには，必ず［改行］キーを押す．このキーの名称は機種によってさまざまで，

［RETURN］

［CR］　　(Carriage return の略)

［↵］

などと表記されている．

NEW　新規にプログラムを入力するときには，まず

NEW ［改行］

を入れる．これは「新しいプログラムを入れますよ」という，パソコンに対する意志表示である．これを入れると，パソコンは，それまで入っていたプログラム（たとえば，前に使った人が残していったもの）を消して，新規入力のための準備を行なう．なお，電源を入れて最初に使うときは，NEWを入れなくてもよい．

AUTO　次に

AUTŌ ［改行］

を入れる．これは「行番号を自動的に付けて下さい」という意味である．これを入れると，ただちに，最初の行番号

10

が表示され，以後，1行入れるごとに，20，30，40，…というように10間隔の行番号が自動的に表示され，行の先頭に付加される（10番おきになっているのは，あとで説明するプログラムの修正作業を容易にするためである）．なお，最初の行番号や間隔を指定することもできる．その場合は，次のように入れる．

AUTŌ 最初の行番号，間隔

(例)　AUTŌ 3000，1

行の継続　プログラムやデーターの1行がディスプレイの画面の1行におさまらない場合，途中で[改行]キーを押してはいけない（[改行]を押すと，そこが行の終りとみなされてしまう）．画面の右端に来ても，かまわず入力を続ければよいのであって，そうすると画面上では自動的に次の行に表示されるが，入力した文字列としては前の続きとして扱われる．

LIST　プログラムを入れ終ったら

LIST [改行]

によって，いま入力したプログラムを表示させ，正しく入力されているかどうかを調べてみるとよい．

［**表示の中断**］長いプログラムで1画面に表示できない場合，**スクロール**といって，表示内容を1行ずつ上にずらして，一番下の行をあけて，そこに新しい行を表示する．そのため，画面の上の方に表示されている行は，上から1行ずつ画面の外に追い出されてしまう．したがって，途中で停止させないと，ゆっくり画面を見ることができない．LISTを中断するための操作法は機種により異なり，普通は

[ESC]（エスケープ）キーを押す　STOP

[BREAK] キーを押す

スペース・バーを押す

の内のどれかで止まるが，どうやっても止まらない場合は

LIST 表示開始行，表示終端行

（例）LIST 600−800

によって表示範囲を指定する．

［**プリンターへの出力**］プログラム・リストをプリンターに出力する方法は機種によって異なるが，たとえば

LLIST　(PC-8001，PC-8801 など)

LIST/P　(MZ-80)

LIST"LPT0:"　(Level-3，Micro8)

などによって行なうことができる．

2.5 スクリーン・エディター

誤字訂正　入力の誤りに気付いたときは，次のようにして誤りを訂正することができる．

1）行の途中（改行キーを押す前）で誤りに気付いた場合は，カーソル* を誤りの所までもどして正しい文字に直せばよい．

2）改行キーを押したあとでも，まだ誤りの箇所が画面に表示されていれば，カーソルをそこまでもどして正しい文字に直し，改行キーを押す（改行キーを押さないと，画面上では直っていても，メモリーの内容は訂正されない）．

3）誤りの箇所が画面上にないときは，

　　EDIT　訂正したい行の番号　改行

または

　　LIST　訂正したい行の番号　改行

（またはスクロール•キーの付いている機種ではそれを用いてもよい）によって，誤りの部分を画面に表示し，そこにカーソルを合わせて訂正し改行キーを押す．

なお，上記2），3）はプログラムの入力時にのみ有効である．

カーソルの動かし方　キーボードにカーソルを移動するためのキーがあり，移動方向が矢印で示されている．これは

　　[↑]
[←]　[→]　あるいは　[←][→][↑][↓]
　　[↓]

のように4方向独立のキーになっている機種もあり，

[↑↓]　[←→]　あるいは　[↓↑]　[←→]

MZ-80C/K　　　　PC-8001

のようにシフト・キーで向きを切り換える機種もある．

---

* Cursor，画面上で現在の入力可能位置を指している印（点滅している）．

(例1) ABCDEと入力したいとき，誤ってBのかわりにVを押してしまい，直後に気が付いたときは，

AV_ (_はカーソルで点滅している)

と表示されている状態から，[←]キーを1回押して

AV

とし(これでVの位置に入力できる状態になった)，Bのキーを押し，続けて残りの文字CDEを入力すればよい．

(例2)
```
10 A-1
20 B=3.14
30 C=A+B_
```
(_印はカーソル)

まで入力した時点で，最初の行の「-」は「=」の誤り，ということに気が付いたときは，[↑]を2回，[←]を4回押せば

```
10 A-1
```

になるから，ここで「=」を押し，[改行]キーを押し，[↓]と[→]で先ほどの位置までカーソルをもどして，続きの部分を入力すればよい．

文字の挿入 画面に表示されている行の一部に文字を1字だけ挿入するには，次のようにする．

1) 挿入したい位置にカーソルを合わせる
2) [INS] (insert) キーを押す
3) 挿入したい文字のキーを押す

(例3) 10 PRNT を，10 PRINTに直すには，Iを挿入したい位置(Nの所)にカーソルをもってゆき

```
10 PRNT
```

[INS]キーを押し，次にIのキーを押す．

2字以上を挿入する方法は機種によって異なり，

2') 挿入したい文字の数だけ[INS]キーを押す

という方式の機種と，

2") [INS]キーは1度だけ押す(「挿入モード」にする)
3") 挿入したい文字(何字でもよい)を入れる
4") 「挿入モード」を解除する(方法は機種により異なる)

という方式の機種がある．

行の挿入　　BASIC のプログラムは各行に行番号が付いている．行番号は普通，10，20，30，…のように10番おきになっている．あとで新しい行を挿入する必要が起こったら，それらの中間の行番号を付けて入力すればよい．

（例 4）　既に入力したプログラム

```
10 A=1
20 C=A+B
```

の途中にB=2 という文を挿入したい場合，10と20の間の番号（たとえば15）を付けて

15　B=2　[改行]

と入力すればよい（画面の上では行番号10，20，15の順になるが，コンピューターの内部では正しく10と20の間に15が挿入されている．それを確認したければ，後述のLISTというコマンドで清書してみればよい）．

［別法］　拡張 BASIC においては，マルチ・ステートメントといって，一つの行に二つ以上の文を書くことができるので（詳しくは43ページ参照），上の例のような場合

10　A=1_

のところにカーソルを移して：B=2 を入力し

10　A=1：B=2

としてもよい（：印はマルチステートメントの際の文の区切りを表わす）．

削　除　　入力した文字を削除するには，普通，

　削除すべき位置より一字右にカーソルを合わせる

　[DEL]（delete の略）キーを押す*．

とすればよい．

行の削除　　削除したい行の行番号に続けて [改行] キーを押す．たとえば行100を消すには

　　100　[改行]

とすればよい．

---

*　Micro-8 の場合は [BS]（back space）．

コントロール・キー　キーボードには，たいてい，CTRLというキーがある*．これはコントロール・キーといって，単独に押したのでは何の効果もないが，これを文字のキーと同時**に押すことにより，下記のような各種の画面操作を行なうことができる．使用法は機種ごとに異なるので詳しくは各社のマニュアルで研究されたい．

	Level-3	PC-8001	Hu BASIC
停止（注1）	C	C	
カーソル位置から行の終りまでを消去	E	E	E
カーソル位置から画面右下までを消去	Z		L
画面出力の一時停止	S	S	
カーソル位置で2行に分割		J	H
カーソルのある行と上の行をつなぐ			A
カーソルのある行と下の行をつなぐ			B
カーソル位置より上へ1行スクロール			Ō
カーソル位置より下へ1行スクロール			P
カナ→英字変換（例，スナミ→RUN）			ケ
画面コピー			C

（注1）　入出力機器の動作を中断させたいのに，普通の停止キーでは止まらない場合，この方法だと止まることがある．

RENUM　挿入や削除によって不等間隔になった行番号を

RENUM 改行

によって，きれいに付け直すことができる．後述のGŌTŌ文などの行先は，それに合わせて自動的に修正される．

RENUM　新行番号，旧行番号，間隔

という形式で指定すれば，「旧行番号」から先の部分だけを，新行番号から始まる指定間隔の番号に付け替えることができる．

---

*　MZ-80にはこれがないが，HuBASICではBreakキーにより代用して上記と同様な機能を持たせている．

**　実際にはCTRLキーを先に押し，押し続けながら文字のキーを押すのが安全．

2.6 実行の開始，中断，再開

開　始　プログラムの実行を開始するには，キーボードから

RUN [改行]

を入れればよい．キーボード操作に慣れていない人は，画面の下端を見ると，いくつかの単語が表示されていて*，その中に[RUN]というのがある筈だから，それに対応するファンクション・キー（普通，キーボードの最上段に並んでいて，

[F1] [F2] [F3] [F4] [F5]

または

[PF1] [PF2] [PF3] [PF4] [PF5]

などと刻印されている）を押せばよい．

中　断　実行を中断するには

[STŌP]

または（機種によっては）

[BREAK]

というキーがあるから，それを押せば，たいてい止まる（これで止まらない場合の処置については25ページ注1参照）．

再　開　実行再開の指令は

CŌNT [改行]

---

* 機種（または使用状態）によっては表示されていないものもある．参考までに二，三の機種の標準設定を以下に示す．

	PC-8001	Level-3	PC-6001
F1	load"	LŌAD [改行]	CŌLŌR
F2	auto	?DATE$,TIME$ [改行]	CLŌAD"
F3	goto	KEY␣	GŌTŌ
F4	list	LIST [改行]	LIST
F5	run [改行]	RUN	RUN [改行]

3　BASIC 入門――簡単な計算と入出力

3.1　BASIC とは

BASIC は，コンピューターを使うための言語（プログラミング言語）の一種である．ほかにも

FORTRAN　COBOL　PL/I　PASCAL

など各種のプログラミング言語があるが，BASIC には

簡単で覚え易い

という特長があり，そのためパソコンでは広く使われている．

BASIC は，いわばパソコンの母国語である．多くの機種では電源を入れると同時に BASIC をすぐ使える状態に入る．BASIC の使えないパソコンは無いといってよい．パソコンを買えば必ず BASIC が無償で付いてくる．それに対し，他の言語はどの機種でも使えるとは限らず，有料であり，しかも非常に高い．

BASIC は1964年，Dartmouth 大学の J. G. Kemeny と T. E. Kurtz により，学生にコンピューターを教えるための言語として開発され，教育用言語として，また TSS* 用言語として知られていたが，パソコンで使われるようになってから機能は大幅に拡張され，今日では実務にも広く用いられている．

本書では，現在パソコンで使われている拡張 BASIC（機能を大幅に拡張した BASIC）について説明する．本書の記述は，大型コンピューターやポケット電卓の BASIC などには通用しないので注意されたい．

---

*　TSS = time sharing system（たくさんの人がコンピューターを同時に対話的に利用できるようにしたシステム）

3.2 命令とプログラム

直接実行形式　パソコンは命令どおりに動く．たとえば

LET A=2　　（Aの値を2にせよ，という意味）

と命令すると，変数Aの記憶場所を用意し，そこに値2を入れる．また

LET B=3　　（Bの値を3にせよ，という意味）

と命令すると，変数Bの記憶場所を用意し，そこに値3を入れる．次に

LET C=A+B　　（A+Bの値をCに代入せよ）

と命令すると，変数Cの記憶場所を用意し，A+Bすなわち2+3の計算をし，結果5をCの所に入れる．さらに

PRINT C　　（Cの値を表示せよ，という意味）

と命令すると，Cの値が表示される．

［**実習**］ もし手もとにパソコンがあったら，上の例を実際にやってみるとよい．すなわち

```
LET A=2 [改行]
LET B=3 [改行]
LET C=A+B [改行]
PRINT C [改行]
```

と入力してみるとよい．即座に結果5が次の行に表示されるであろう．なお，ついでに

PRINT A,B [改行]　　（AとBの値を表示せよ）

と命令すれば，AとBの値が確かに2と3になっていることを確認することができる．

命令は一定の形式（文法規則）に従って書かなければいけない（そうしないとパソコンの側で解読できず，実行できない)．その規則を覚えることがパソコン活用の第一歩である．

本章では BASIC における最も基本的な命令

LET IF GŌTŌ FŌR NEXT DIM
INPUT PRINT DATA REM END

について説明する（ただし変数の型や文字列処理に関する事項は除く）．

間接実行形式（普通の使い方）　左記の例のように，1ステップごとに使用者が命令する方式では，複雑な計算はできない．そこで普通は一連の命令（これをプログラムという）をあらかじめコンピューターに記憶させ，命令を順にとり出してきて実行する，という方式（プログラム記憶方式，間接実行形式）を用いる．

BASICでこれを行なうには，命令の頭に行番号を付けて入力し，プログラムを入れ終ったら

RUN　改行

というコマンド（直接実行命令の一種）で実行を指令すればよい．

> **行番号**　行番号としては1～9999（0を許す機種や32000あるいは65000ぐらいまでを許す機種も少なくないが）の範囲内の整数値を用いることができる．
>
> プログラムは原則として行番号の順に実行される．行番号はまた，プログラムの修正，挿入，削除の際の，位置の指定に用いられる（24ページ参照）．行番号は連続した番号でなくてもよい（欠番があってもよい）．そこで普通は
>
> 10，　20，　30，　40，　50，
>
> というように10とびの番号を付ける*．

［**実習**］左記の例を間接実行形式で実行させてみよう．それには

```
10 LET A=2 改行
20 LET B=3 改行
30 LET C=A+B 改行
40 PRINT C 改行
RUN 改行
```

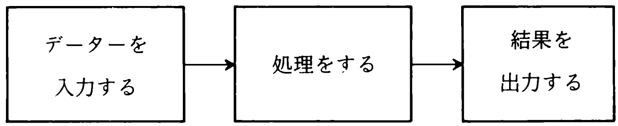
と入力すればよい．行40を入れた時には何も出力されず，RUNさせてはじめて結果5が表示されるであろう．

---

*　AUTOというコマンドで「自動番号付加」を指令しておくと，自動的にこのような番号を付けてくれる．20ページ参照．

3.3　INPUT 文

プログラムは，普通，



という構造をしている．本章では，これに従って

　　入力の部分の書き方

　　処理の部分の書き方

　　出力の部分の書き方

の順に文法を説明することにする．

INPUT 文の基本形　　データー入力のためにはＩＮＰＵＴという命令がある．最も単純な書き方は

	ＩＮＰＵＴ　変数名
(例)	ＩＮＰＵＴ　Ａ

で，その変数の値を入力せよ，という意味になる．二つ以上の変数の値を入力するときは，変数名をコンマで区切って書く．

(例)　ＩＮＰＵＴ　Ｘ，Ｙ，Ｚ

実行時の操作方法　　ＩＮＰＵＴ文を実行すると，入力要求を表す？印が表示される*ので，キーボードから値を入れる．数値には，必要に応じて，符号，小数点，指数部などを付けることができる．一般形は

　　±整数部．小数部Ｅ±指数部
　　↑（符号）　　　　↑（指数部の符号）

で，指数部Ｅ±nは「$\times 10^{\pm n}$」を表す．

(例)　－１Ｅ６　は　-1×10^{6}

　　　３．２Ｅ－１　は　3.2×10^{-1}

を表す．

---

*　？印の出ない機種もあるが，カーソルが出るのでわかる（処理中はカーソルが消えている）．

二つ以上のデーターを入れる場合はコンマで区切る．一つのINPUT文)に対する入力値は，原則として1行で入れなければいけない*．データーの箇数が合わないと，補充入力要求(たとえばPC-8001の場合だと??印）または入れ直し要求（たとえばLevel-3の場合だと

```
Redo From Start
```

というメッセージ）が出る．

入力要求メッセージの出し方　上記の事情を考えると，入力の際

今，何のデーターを入れたらよいか

どこまでを1行に入れなければいけないか

などの情報を画面に表示することが望ましい（そうしなければ一般の利用者には使えない)．BASICには，そのためのメッセージを表示する機能がある．書き方は次のとおり．

```
       INPUT "メッセージ"; 変数名, …, 変数名
(例)   INPUT "A,B="; A,B
       INPUT "xをいれてくだ さい"; X
       INPUT "あなたは なんさいて すか"; N
```

このようなINPUT文を実行すると，まずメッセージが表示され，その右に?印が表示され，その右にカーソルが出る．なお，機種によっては**?印の表示を出さないことができる．それには

```
INPUT "メッセージ", 変数名, …, 変数名
```

と書けばよい（;印のかわりに,印を用いる)．

その他の方法　データーを入れるための命令としては，READ文というのもある(後述)．また，1～2回しか使わないプログラムであれば，前節の例のように，代入文（LET文）で書くのも簡単でよい．

---

*　画面では何行かにまたがってもよい．画面の方は右端に来ると自動的に次の行に移ってくれるから，かまわず続けて入力する．文法上は「改行を入れるまでが1行」として扱われる．

**　たとえばLevel-3やMicro-8

3.4 代入文（LET 文）

> 代入文の正式の書き方は
>
> LET　変数名＝式
>
> である．しかし大部分の機種ではLETを省くことが許されているので
>
> 変数名＝式
>
> という形で書くのが普通である*．
>
> (例)　LET　C＝A＋B
>
> C＝A＋B

これらの文は，「＝の右に書かれた式の値を計算して，＝の左に書かれている変数に代入せよ」という意味になる．

(例)　Aの値が2，Bの値が3のとき，代入文

C＝A＋B

を実行すればCの値は5になる．

［**備考**］ 1）代入先の変数名は必ず＝の左側に，式は必ず＝の右側に書かなければいけない．したがって，次のような書き方は許されない．

LET　A＋B＝C

2）単独の変数または定数も，広い意味での「式」の一種とみなされる．したがって

LET　A＝1

などと書いてもよい（「1」だけでも「式」である）．

3）変数の値は代入文によって何度でも書き替えることができる．その特別な場合として，左辺の変数名が右辺の式に現れてもよい．その場合，式の値を計算する段階では代入前の値が使われ，その結果が左辺の変数の新しい値として代入される（書き替えられる）．

(例)　Iの値が2のとき，代入文

I＝I＋1

を実行すると，まずI＋1（すなわち2＋1）の値が計算され，その結果3がIに代入されるから，実行後のIの値は3になる．要するに，この代入文は「Iの値を一つふやせ」という命令だと思えばよい．

---

*　現在わが国で広く市販されているパソコンの内，LETの省略を許さないのはシンクレアのZX-81だけであろう．

3.5 変 数 名

■ 変数名としては次のような名前を用いることができる.

1）英字1字 —— これはすべての機種で使用できる.

2）英字1字の次に数字1字を付けたもの —— 一部のポケット・パソコンを除くすべての機種で使用できる.

(例) V9

3）英字2字（ただし，IF，TŌ，ŌNなど，BASICで特別な意味をもつ単語を除く）—— 大部分の機種で使用できる*.

(例) CQ

4）先頭は英字，第2字以後は英字または数字(ただしBASICで特別な意味をもつ単語をその綴りの一部に含んではいけない)

(正しい例) Z80A

(誤りの例) DIF （IFを含むからいけない)

—— これも大部分の機種で利用できるが,「識別には先頭の2文字のみが有効」いいかえれば「先頭2文字が同じなら，あとが違っていても同じ変数とみなす」という扱いをする機種がかなりあるので注意を要する**.

(例) そういう機種ではXMAXとXMINを区別できない.

5）先頭は英字，第2字以後は英字または数字またはピリオド（綴りは自由）—— これは一部の機種でしか使えない***.

■ 記号，カナ文字，漢字などは，普通，変数名に使えない****. 小文字も事実上，使用できない（小文字で入力しても内部で大文字に変換されてしまうのが普通である).

---

* 3）を許さない機種の例： MB6885

** 「先頭2文字のみ有効」という扱いの機種の例： VIC1001，PC-6001，PC-8001，MZ-80 B/C/K，SORD M223，APPLE II

*** 本書の執筆時点ではPC-8801のみ.

**** PASOPIAのOA-BASICはカナ文字の変数名を許す.

3.6 式の書き方

演算子　加減乗除を

＋　－　＊　／

で表す．ベキ乗は^印で表す*

(例)　X^Y　は　X^Y　で表す．

カッコ　丸カッコ（　）だけが使用可能．何重に使ってもよい．

(例)　$\dfrac{1}{x+\dfrac{1}{x+1}}$　は　1／(X+1／(X+1))　で表す．

計算順序　普通の数式と同様に

カッコの中を先に計算する

加減算よりも乗除算を先に計算する

さらにベキ乗があれば先に計算する

という規則で処理される．なお優先順位が同じものについては

左から順に計算される

のが普通である．

(例)　A／B／C　は　(A／B)／C　と同じ結果になる．

[**備考**]　上記以外の記号，たとえば

!　%　#　$　¥

などは，普通の数式における意味（階乗，パーセントなど）には使えない．特に注意を要するのは！および%で，これは後述のようにそれぞれ実数型および整数型を表す属性文字であるから，たとえば

5!　5%

は「5の階乗」「5パーセント」ではなく「整数型の5」「実数型の5」を表す．

整除（商が整数で表せる所まで割る）は¥印，**余り**はMŌDで表す．

(例)　C=A¥B　　AをBで整除した商をCとする．

　　　R=P MŌD 2　　Pを2で割った余りをRとする．

＊　ただしMZ-80では↑印で表す．(例)　X↑Y

3.7 組込み関数

BASIC では次の関数を使用できる（注 1）.

種類	数学記号	BASIC における書き方	
平方根	$\sqrt{x}$	SQR(引数)	
指数関数	e^x	EXP(引数)	
自然対数	$\log_e x$, $\ln x$	LŌG(引数)	(注 2)
正弦	$\sin x$	SIN(引数)	(注 3)
余弦	$\cos x$	CŌS(引数)	(注 3)
正接	$\tan x$	TAN(引数)	(注 3)
逆正接	$\arctan x$	ATN(引数)	(注 4)
絶対値	$\|x\|$	ABS(引数)	
整数化	$[x]$	INT(引数)	(注 5)
符号		SGN(引数)	(注 6)

注 1）このほかに，

文字列処理のための関数

乱数発生のための関数

状態を調べるための関数

などがある.

注 2）自然対数を LN(X) で表し，常用対数を LŌG(X) とする機種もある（たとえば MZ-80B/C/K, SORD M-23 など）.

注 3）引数の単位はラジアンとして扱う．度をラジアンに直すには

$\pi/180 \fallingdotseq 0.0174533$

を掛ければよい.

注 4）単位はラジアン．値の範囲は $-\pi/2$〜$\pi/2$ になる.

注 5）x を越えない最大の整数.

(例) INT(3.14) は 3　　INT(−3.14) は −4 になる.

注 6）$x<0$ ならば SGN(X) の値は −1 になる

$x=0$ ならば 〃 0 〃

$x>0$ ならば 〃 +1 〃

3.8 PRINT文

結果を表示するにはPRINT文を用いる．最も簡単な使用法は

　　PRINT　変数名

である（28ページの例参照）が，一般に式を書いてもよい．

　　PRINT　式

（例）　PRINT　A+B

二つ以上，列記したい場合は，コンマまたはセミコロンで区切って書く．

（例）　PRINT　A,B,C+D

　　PRINT　A;B;C+D

セミコロンで区切った場合は，前の値にすぐ続けて（間をあけずに*）次の値が出力される．それに対しコンマで区切った場合は，一定の位置までスキップしてから次の値が出力される．

```
10 A=12.3
20 B=5.67
30 C=-980
40 PRINT A,B,C
50 PRINT                 これは間を1行あけるため
60 PRINT A;B;C
```

の実行結果は次のようになる．

```
12.3            5.67            -890

12.3  5.67 -890
```

（上はコンマで区切った場合，下はセミコロンで区切った場合である）

＊　ただし数値データーの出力の場合，1字分の空白が入る．また先頭は符号であるが，＋印のかわりに空白が出力される．そのため正数の前には2字分の空白が入ることになる（上の例の12.3と5.67の間がそれである）．

＊＊　位置は機種により異なるが，普通，15字ぐらいの間隔で設定されている．

改行と非改行　ＰＲＩＮＴ文と改行の関係は，「特に指定しない限り，１箇のＰＲＩＮＴ文が出力の１行に対応する」というのが第１原則である．

（例）　ＡとＢとＣを１行目に，ＤとＥを２行目に出力したければ

```
PRINT A,B,C
PRINT D,E
```

とすればよい．

> 同じ行に続きを出力したい場合は，ＰＲＩＮＴ文の最後にコンマまたはセミコロンを書いておく．左のページで述べた規則と同様に，コンマならば一定位置までスキップし，セミコロンならば間をあけずに（ただし前ページの脚注* 参照）続きが出力される．

（例）

```
PRINT A,
PRINT B
```

は　ＰＲＩＮＴ　Ａ，Ｂ　と同等．

逆に一つのＰＲＩＮＴ文の途中で改行させることは，不可能ではないが*，プログラムの明快さを損ねるので，あまり使わない方がよい．

なお，以上のようにして定まる１行の文字数が画面の幅（１行に表示できる文字数）をオーバーする場合には，はみ出す直前の数値の区切りの所で自動的に改行される．

> **見出しの付け方**　見出しなどのために文字列を出力したい場合は，出力したい文字列を二重引用符（"印）で囲んで，ＰＲＩＮＴ文の出力項目の所に書く．

（例）　ＰＲＩＮＴ　"きょりは"；Ａ；"km"

その出力例（Ａの値が１０１．６の場合）

```
きょりは 101.6 km
```

出力される数値の書式　符号や小数点を含めて８桁ぐらいでおさまる数値はそのままの（普通の）形で出力され，それでおさまらない数値は指数部付きの形（±△．△△△△△Ｅ±△△）で出力される．

＊　出力項目として改行コードＣＨＲ＄（１３）を書けばよい．

3.9* PRINT USING と TAB

前節で説明した出力方法は，出力書式の指定が不要なので簡単であるが，場合によっては「もっと手間をかけてもよいから，きめのこまかい書式指定をして，こちらの思いどおりの形で出力したい」ということもある．そのような場合は PRINT USING文 とTABを用いるとよい．

PRINT USING 文　これは書式指定付きの出力文で，次の要領で書く．

(例)　PRINT USING"+##.### ";A;B

これは，「AとBを

　　符号は数値の前に付ける．正は+，負は−で表す
　　整数部に2桁とる
　　左から4字目に小数点を打つ
　　小数部に3桁とる
　　その右に1字分の空白を入れる

という形で出力せよ」ということを表す

[**備考**] 1）符号の位置に「#」を書くと，正の場合の符号表示が+印でなく空白になる．

2）符号を右に付けることもできる．それは+か−を右に書くことによって指定する．「−」印は上記1）と同様，「+サプレス」の意味．

(例)　PRINT USING"###.##+";X

3）整数値として右づめで表示したい場合は「.」印は不要．

(例)　PRINT USING"−###";N

4）3桁ごとにコンマを入れることも可能．

(例) PRINT USING"+#,###,###";KINGAKU

5）指数部付きの形（浮動小数点表示）で出力したい場合は，右に
　　^^^^
を付ける．指数部はE±××の形で出力されるので常に4桁必要である．

---

*　初心者はここをとばして先に進んでよい．

[プログラム例と出力例]

```
10 A=234.56
20 PRINT USING"+###.##";A
30 PRINT USING"####.##";A
40 PRINT USING"+####.#";A
50 PRINT USING"#####.#";A
60 PRINT USING"+######";A
70 PRINT USING"#######";A
80 PRINT USING"+#.####";A
RUN
```

```
+234.56
 234.56
 +234.6
  234.6
   +235
    235
%+234.5600
```

} MB-6890 による出力例

最後の%印は桁数オーバーのため指定書式で出力できないことを表す.

TAB　タイプライターにTABというキーがある. tabulate すなわち作表のための機能で, このキーを押すと, あらかじめ指定してあった文字位置までスキップする. BASIC のTABも機能は同様で, PRINT文の出力項目の一つとしてTAB(*n*)と書いておくと, 第*n*文字位置(行の左端を0として数える)までスキップし, 次の出力項目はそこから出力される. ただし, 現在位置より左にもどることはできない.

(例)

```
10 PRINT "01234567890123456789"
20 A=-753
30 PRINT A;TAB(12);A
40 PRINT TAB(3);A;TAB(12);A
50 PRINT TAB(5);A;TAB(12);A
RUN
```

```
01234567890123456789
-753        -753
   -753     -753
     -753   -753
```

3.10 READ文とDATA文

これまでのプログラム例では，データーの入力方法として，

入力の指示をINPUT文で書く

実際のデーターは実行時点に入れる

という方法をとってきたが，もう一つの方法として

データーをプログラムと一緒に入れておく

その入力指示をREAD文で書く

ということができる．すなわち，入力すべきデーターを

DATA　1番目のデーター，2番目のデーター，…

（これで書ききれなければ，同じ形式で何枚でも並べる）

の形でプログラムの中（普通はプログラムの最後の所）に書き，それをREAD文で読む．READ文の書き方はINPUT文と同じ形

READ　変数名，変数名，…

である．INPUT文とREAD文の違いは

INPUT文は，実行時に端末機からデーターを読む

READ文は，DATA文で書いたデーターを読む

という点にある．DATA文とREAD文の組合せによる方法には次のような利点がある．

1）統計の計算などの場合，同じデーターをいろいろな手法で解析したいことがある．そういうとき同じデーターを何度も入力しないで済む．

2）デバッグのとき，テスト・データーを何度も入れるのはめんどうであるが，この方式なら一度入れるだけでよい．

3）この方式だと，RESTŌRE文を用いて「いまのデーターの最初の位置にもどれ」という指示ができる．こうして，一つのプログラムの中で，同じデーターを何回も読むことができる．

要するに，INPUT文で入力したデーターは，すぐ消えてしまってあとに残らないのに対し，DATA文で書いたデーターはプログラムの一部として記憶装置に残っているので，同じデーターを何度も利用することができるわけである．

3.11 REM, STOP, END

プログラムの中に注釈（説明，覚え書き）を書いておくと，あとで読むとき，わかり易くてよい．注釈はREM文の形で書く*．書き方は

REM 注釈

で，注釈としては任意の文字列（英数字，カナ文字，記号，空白など，何でも自由に使用できる．ただし 改行 を除く.）を書くことができる．

REM文はプログラムの中のどこに書いてもよい．ただし（後述の）マルチ ステートメントに関しては注意が必要である（次項参照)．

(例) プログラムの冒頭にはタイトルその他を書いておくとよい．

```
10 REM *******************
20 REM * ほうていしき f (x)=0をとく *
30 REM *    (ニュートンほう)    *
40 REM *     version 1,1     *
50 REM *******************
```

STŌPとENDは，読んで字のごとく，STŌPは「止まれ」，ENDは「実行終了」を意味する．STŌPは一時停止であって，コマンド

CŌNT 改行

を入れればSTŌPの次の文から再開（続きを実行）させることができる．それに対し，ENDは終点であって，その続きは無い．違いはそれだけである**．

---

* REMは remark の意味.

** FORTRAN においては，ENDが「プログラムの最後の行」という特別な意味をもっているが，BASIC のENDはそのような意味をもたない．したがって，ENDを何箇所に書いてもよく，一つも書かなくてもよい．これまでに例示したプログラムのように，STŌPもENDも書かなくても，行番号最大の文に来れば自然に実行終了となる．

3.12 文法細則

以下で説明する事項は，知らなければ知らなくても済むことであるが，覚えておけば役に立つ．

> **変数の初期値は 0**　プログラムの実行開始時点において，すべての変数の値は 0 になっている．

（例）何も代入しないで　PRINT A　を実行すれば0が出力される．

> **文中の空白は無視***　本書では，プログラムを読み易くするため，文の中に空白をたくさん入れているが，BASIC では（REM文および2重引用符で囲まれた部分を除き）空白はすべて無視される（したがって，空白は入れても入れなくてもよい）ということになっている*．

（例）
```
10 INPUT A,B   と  10INPUTA,B   は同じ
20 LET C=A+B   と  20LETC=A+B   は同じ
30 PRINT C     と  30PRINTC     は同じ
```

> **省略記法**　記号？と ' は命令の略記法に用いられ，
>
> ?　印は　PRINT
>
> '　印は　REM
>
> を表す．

［**解説**］？印は直接実行形式の場合によく用いられる．たとえば，変数Aの内容（現在記憶されている値）を調べたいときに

? A 改行

とやって表示させたり，$\pi/3$ の値を知りたいときに，電卓がわりに

? 3.1416/3

というぐあいに用いたりする．一方，REMのかわりに ' を用いるのは，英語やローマ字で注釈を書いた場合，REMが注釈の文章の一部のように見えることがあるので，それを避けるため，というのが一つの理由であり，また ' 注釈 ' という形で書けば，いかにも注釈らしく見える，という理由もある．

---

* ただし PC-8801 は例外。

マルチ・ステートメント

一つの行に二つ以上の文を書くことができる．これをマルチ・ステートメントという．二つ以上の文を書く場合は，文の間をコロンで区切る．

(例) 10 INPUT A,B : PRINT A+B

行番号　一つの文　区切り記号　もう一つの文

[解説] 本書では，これまで常に，一つの行に一つの文を書いてきた．それが伝統的なBASICの標準的な書き方である．1行1文ならば，すべての文に行番号が付いており，それによって編集したり，行先を指示（5章で説明する）したりすることができる．初心者はなるべくそういう形で書く方がよいと思う．

しかし，少し慣れてきて，ある程度長いプログラムを作るようになると，1行にいくつもの文を書きたくなる．それは次のような理由による．

1）ディスプレイの1画面になるべくたくさんの文を表示できる方がデバッグをし易い．

2）処理の内容によっては，横に並べて書く方が自然な場合がある．たとえば，複素数の計算をするとき，実数部の処理と虚数部の処理を並行して書くと見易い．

3）他のプログラミング言語ならば一つの文で書けるのにBASICだと複数箇の文でないと書けないことがある．そういうとき，一つの機能的まとまりを示すために1行で書くと見易くなる．

4）1行1文にするよりも，マルチ・ステートメントにする方が，記憶場所をいくらか節約できる．

5）文の挿入の一手段として使える

[注意] REM文と後述のIF文に関しては「行の終りまでを一つの文とみなす」という扱いになっているので，REM文やIFの右にマルチ・ステートメントの形で他の文を書くことはできない．

(例) 10 REM "example" : INPUT A

の「INPUT A」は注釈の一部とみなされる．

3.13 例題および演習

例題 1 ―― 和差積商

二つの数値を読み込み，その和，差，積，商を計算して表示するプログラムを作れ。

［**解答**］ 和，差，積，商を変数名WA，SA，SEKI，SYŌUで表すことにすれば，プログラムは次のようになる。

```
10 INPUT A,B
20 WA=A+B
30 SA=A-B
40 SEKI=A*B
50 SYŌU=A/B
60 PRINT WA,SA,SEKI,SYŌU
```

［**別解**］ PRINT文に式を書くことができるので，それを利用して次のように書いてもよい。

```
10 INPUT A,B
60 PRINT A+B,A-B,A*B,A/B
```

［**別解**］ 入力要求表示や結果の見出しを付ければ，さらに使い易いプログラムになる。

```
10 INPUT "A=";A
20 INPUT "B=";B
30 PRINT "A+B=";A+B
40 PRINT "A-B=";A-B
50 PRINT "A*B=";A*B
60 PRINT "A/B=";A/B
```

［**解説**］ 二重引用符で囲まれた部分は，入力要求表示および見出しのための文字列である。それに続くセミコロンは「間をあけずにすぐ続けて表示する」ことの指示である。

例題 2 ———— 円の面積

> 半径 r の値を読み込み，円の面積 $S=\pi r^2$ を計算して出力するプログラムを作れ

［**解答**］ π という文字またはPIという綴りが円周率を表す機種もあるが(たとえばMZ-80，M-23 など)，一般的には π の値を数値として書く必要がある．パソコン内部の演算桁数は，普通，約 6 桁であるから3.14159あたりまで書けば十分である．

```
10 INPUT R
20 S=3.14159*R^2
30 PRINT S
```

［**別解**］ 入力要求表示，結果の見出しを付けると次のようになる．

```
10 INPUT "はんけい(cm)=";R
20 S=3.14159*R^2
30 PRINT "めんせき=";S"cm^2"
```

［**別解**］ π を使える機種では，大いにそれを活用するとよい．以下はMZ-80 のためのプログラム例である．

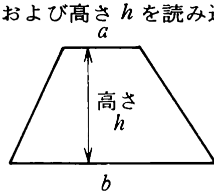
```
10 INPUT "R=";R
20 PRINT "S=";π*R↑2
```

問　題

3.1 （台形の面積） 台形の底辺の長さ a，b および高さ h を読み込み，面積

$$S=\frac{1}{2}(a+b)h$$

を計算して出力するプログラムを作れ．

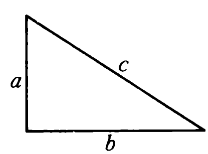


3.2 （ピタゴラスの定理，三平方の定理）

直角三角形の正角をはさむ 2 辺の長さ a，b を読み込み，残る 1 辺の長さ

$$c=\sqrt{a^2+b^2}$$

を計算して出力するプログラムを作れ．



3.3 （三角関数） $\theta = 30°$ に対する

$$\sin\theta \qquad \cos\theta \qquad \tan\theta$$

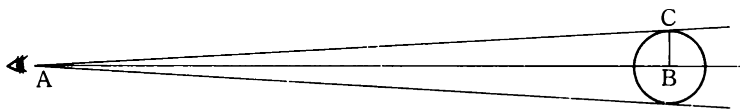
の値を計算して出力するプログラムを作れ．もしもパソコンを実際に使用できるなら，上のプログラムを実行し，厳密な値

$$\frac{1}{2} \qquad \frac{\sqrt{3}}{2} \qquad \frac{\sqrt{3}}{3}$$

（または電卓によって求めた結果）と比較せよ．

［ヒント］ 度からラジアンへの変換を忘れるな．

3.4 （視半径） 地球から月までは約 384400 km，月の半径は約 1738 km である．月の視半径（下の図の角 CAB）θ を求めよ．

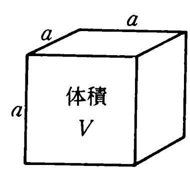


［ヒント］ $\overline{AB}$ と $\overline{BC}$ が既知であるから，簡単のため角 ABC が直角と考えれば*，計算式は次のようになる．

$$\tan\theta = \frac{1738}{384400} \quad したがって \quad \theta = \arctan\frac{1738}{384400}$$

3.5 （立方根） 体積 V の立方体の 1 辺の長さ a を計算するプログラムを作れ．

［ヒント］ $a = \sqrt[3]{V} = V^{(1/3)}$ を計算して出力すればよい（BASIC のべキ乗演算の指数部は整数でなくてもよく，そこに式を書いてもよい）．



＊ 厳密にいえば角 ABC は直角でなく，角 ACB の方が直角であるから，arc sin を用いるべきであるが，結果は大差ない．その差を問題にするなら「地球上のどこで見るか」も考慮する必要がある．なお，arc tan の計算方法は156ページ参照．

4 制御文——枝分かれとくりかえし

4.1 概　　説

BASIC の文は原則として行番号の順に実行されるが，

ある行までスキップしたい（または，ある行にもどりたい）

条件に従って分岐し，それぞれ別の処理をしたい

プログラムの一部分をくりかえし実行したい

という場合がある．そのために，BASIC には次のような文がある．

まず，指定した行に進む文としては

GŌTŌ文………………（行きっぱなし）

GŌSUB文………………行ってもどってくる

分岐に関しては

IF文……………………条件式によって分岐

ŌN文……………………番号によって分岐

くりかえしに関しては

FŌR～NEXT文…………………一定回数の反復

REPEAT～UNTIL文…………条件が成立するまで反復

WHILE～WEND文……………条件が成立している間，反復

がある．以上の内，

GŌTŌ　GŌSUB～RETURN　IF　FŌR～NEXT

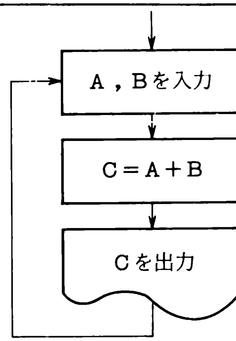
が基本的で，すべての機種で使用できる．それ以外の文は，主として，プログラムの構造を単純にし，意味を明快にする目的で設けられたものであって，それを使用できない機種もあるが容易に他の文で代行することができる．

4.2 GOTO 文とラベル

無条件に特定の行までスキップしたり，もどったりするにはGŌTŌ文（英語の“Go to ～.”の意味であるから［goutu:］と読む．BASICではGŌとTŌの間に空白を入れないのが普通）を用いる．

書き方は GŌTŌ　行先の行番号 (例)　GŌTŌ　100　　「行番号100の文に行け」という意味

(例)　プログラムの最後の所から最初にもどるようにGŌTŌ文を書いておけば，停止ボタンを押すまで何回でも反復実行できる．



(例)
```
10 INPUT "A=";A
20 INPUT "B=";B
30 C=A+B
40 PRINT "A+B=";C
50 GŌTŌ 10
```

ラベル　機種によっては，行先を，行番号でなしにラベルで指定することができる．その場合は，行先の行にラベルを付けておいて

GŌTŌ　ラベル

と書けばよい．ラベルの書き方は機種によって異なる．

	N_{88}BASIC	HuBASIC
ラベルの定義	＊文字列	LABEL"文字列"
GŌTŌ文	GŌTŌ　＊文字列	GŌTŌ　"文字列"
備考		字数255まで可 文字列変数でもよい
(例)	GŌTŌ　＊USA …………… ＊USA	GŌTŌ　"USA" …………… LABEL"USA"

4.3 GOSUB～RETURN

これは，行って処理を行なったあと，もとの所にもどってきて続きを実行したい場合に用いる文で，次のように書く*.

GŌSUB　行先の行番号

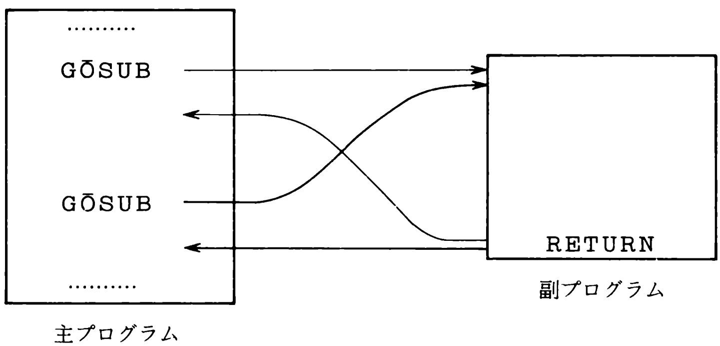
ラベルを使用できる機種ならば次のように書くこともできる.

GŌSUB　行先のラベル

この文を実行すると，行番号またはラベルで指定された所へ飛び，そこのプログラムを実行し，

RETURN

という文に達すると，もとの所（正確にいえばGŌSUB文の次の文）にもどる.



［**備考**］ 他のプログラミング言語（たとえばFORTRANやPASCAL）と違って，BASICの副プログラムは変数，行番号などが主プログラムと全部共通であり，一部の機種以外では，引数を書くことができない．他の言語の副プログラムと同程度の機能が必要な場合には，ディスクBASICのチェイン・ジョブの形を用いるとよい.

---

* SUBはsubroutineの意.

4.4 IF文による分岐

ある条件のときだけ，指定した所に進むようにするには，次のように書く．

　　　IF　条件式　THEN　行先

これを次のように書くこともできる*．

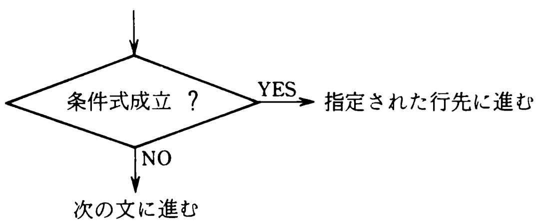
　　　IF　条件式　GŌTŌ　行先

条件が成立しない場合の行先を指定することもできる**．書き方は

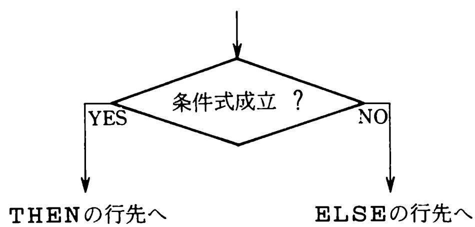
　　　IF　条件式　THEN　行先　ELSE　行先

である．行先は行番号またはラベル(ラベルを使用できる機種の場合)で指定する．

IF～THEN，IF～GŌTŌ



IF～THEN～ELSE



(例)　IF A=B GŌTŌ 100
　　　IF C<D THEN 300 ELSE 500

---

* PC-6001などの小型パソコンを除けばだいたい使用できるようである．

** これも上記*と同様であるが，MZ-80B/C/KはHuBASICではないと使えない．

4.5 条件式の書き方

■ 条件式としては次のようなものが書ける。

1）二つの式を等号または不等号で結んだもの。

(例) A=B M+1>N ABS(E)<0.01

ただし，等号，不等号は次のように書く。

	普通の数学記号	BASIC の書き方*	
等しい	=	=	
等しくない	≠	<>	><
より小さい	<	<	
等しいかまたは小さい	≦	<= または	=<
等しいかまたは大きい	≧	>= または	=>
より大きい	>	>	

2）等式または不等式（すなわち上記1の形）に（必要ならば）NŌTを付け，それらをANDやŌRで結んだもの**。ただし，

AND	は	「かつ」	両方とも成立することを要求
ŌR	は	「または」	少なくとも一方が成立すること
NŌT	は	「でない」	否定

を表す。

(例) A=0 ŌR B=0

3）上記2)の形をカッコで囲み，(必要ならば)NŌTを付け，それらをANDやŌRで結んだもの。なお，カッコを省略した場合には

NŌT AND ŌR

の順に結合される。

(例) NŌT A ŌR B AND C は次式と同等

(NŌT A) ŌR (B AND C)

---

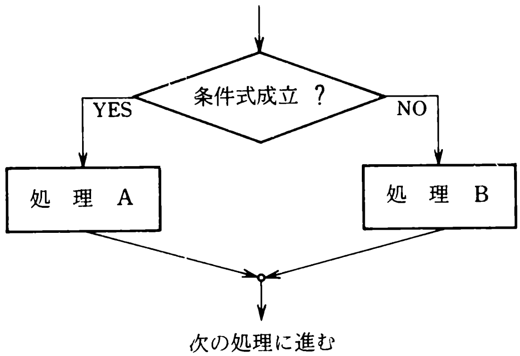
* 左が正式の書き方であるが右のように書くことを許す機種も多い。

** 一部の機種では使用できない。

4.6 IF文による場合分け処理

（この形を使用できない機種もある.）

プログラムの流れには，枝分かれして，それぞれの処理を行なって，また合流する，という形がよくある.



そのような場合，処理が比較的簡単（短いプログラムで書ける）ならばそれをＩＦ文の中に書くことができる．書き方は，

　　ＩＦ　条件式　ＴＨＥＮ　処理Ａ　ＥＬＳＥ　処理Ｂ

またはその後半（ＥＬＳＥ以後）を除いた形

　　ＩＦ　条件式　ＴＨＥＮ　処理Ａ

である．ここで処理Ａと書いたのは「条件式が成立したときに実行すべきプログラム」，処理Ｂと書いたのは「条件式が成立しなかったときに実行すべきプログラム」のことで，実際には

　　一つの文

または

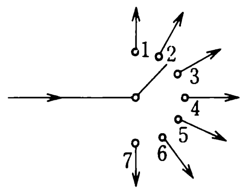
　　いくつかの文をコロン（：印）で区切って１行に書いたもの

の形で記述する.

（例）
```
IF X<0 THEN Y=0 ELSE Y=X
IF N=1 THEN I=0 : J=0 : K=0
IF A<B THEN PRINT A:ELSE PRINT B
```

4.7 ON～GOTO, ON～GOSUB

一度にたくさんの枝に分岐したいことがある.



番号をキー（選択の基準）としてこのような枝分かれを行なうには,

ŌN　番号　GŌTŌ　行先$_1$，行先$_2$，…，行先$_n$

という文を用いると便利である. これは

番号が 1 ならば行先$_1$に行け

2　　行先$_2$

⋮　　⋮

n　　行先$_n$

ということを表している. 行先は行番号またはラベル（ラベルを使用できる機種の場合）で指定する.

(例)　ŌN K GŌTŌ 100,300,1750

番号の所には式を書いてもよい.

(例)　ŌN N−40 GŌTŌ 200,280,170

式の値が整数にならなかった場合には四捨五入により整数化される. その結果が 1～*n* 以外になった場合(いいかえれば, 該当する行先が無い場合）は「次の文に進む」という扱いにしている機種が多いようである.

同様に

ŌN　番号　GŌSUB　行先$_1$，行先$_2$，…，行先$_n$

という書き方もできる. この場合は, 番号で選ばれた行先に進み, そこのプログラムを実行し, RETURN文に到達するとそこからもとの所（ŌN文の次の文）にもどってくる.

4.8 FOR～NEXT

一定回数のくりかえしを行なうにはFŌR文を用いる．これは次の形式で書く．

FŌR 制御変数名＝始値 TŌ 終値 STEP 増分

これは「制御変数の値を始値から終値まで増分ずつ変えて，以下の部分をくりかえし実行せよ」という意味になる．増分1の場合は省略して

FŌR 制御変数名＝始値 TŌ 終値

と書いてもよい．くりかえす部分の最後には

NEXT 制御変数名

を書く．これは「制御変数の値を次の値（すなわち，現在の値プラス増分）に変えて，くりかえし部分の最初にもどれ．ただし，制御変数の新しい値が終値を越えたならば，くりかえしを終了してNEXTの次の文に進め」ということを表す．なお

始値，終値，増分は必ずしも整数でなくてもよい*

始値，終値，増分として変数や式を書いてもよい

増分は負でもよい

（例） N の値を1から5まで変えて $N(N+1)/2$ の値を計算し出力するには次のように書けばよい．

```
10 FŌR N=1 TŌ 5
20     PRINT N,N*(N+1)/2
30 NEXT N
```

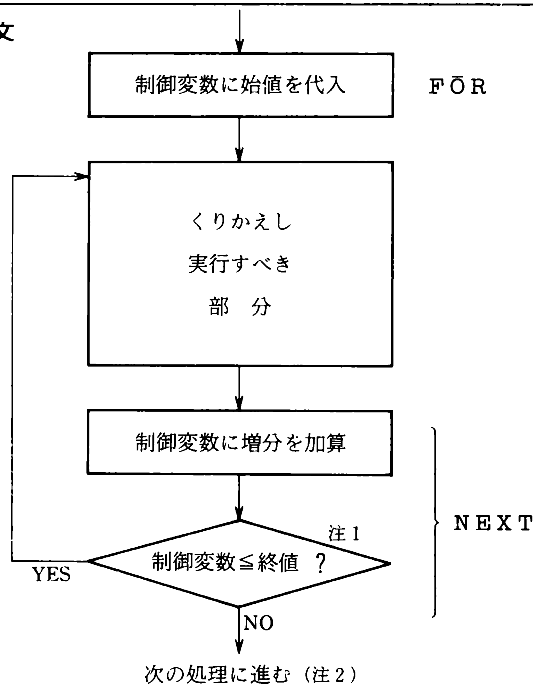
［出力例］

```
1    1
2    3
3    6
4    10
5    15
```

---

* ただし整数でないと，丸め誤差のために，くりかえし回数が，意図したとおりにならないことがあるので，なるべく整数を用いる方が安全である．

FŌR〜NEXT文
概念図*



注 1） 増分が負の場合には，終了判定の不等式の向きが逆になる．

注 2） このとき制御変数の値は，「最後に実行したときの値プラス増分」になっているのが普通である．

多重のくりかえし　FŌR〜NEXTを入れ子の形で書くことにより 2 重，3 重，…の組合せ的くりかえしを行なうことができる．

(例)　I の値を 3 から 5 まで変え，その各 I について，J の値を 1 から I−1 まで変えて J／I を計算し出力するには次のようにすればよい．

```
10 FŌR I=3 TŌ 5
20     FŌR J=1 TŌ I-1
30         C=J/I
40         PRINT I;J;C
50     NEXT J
60 NEXT I
```

内側のループ（20〜50）　外側のループ（10〜60）

---

* 終了判定を，くりかえし部分の前に行なっている機種もある（たとえば PC-8801）が，普通の使い方ならばどちらでも同じ結果になる．

4.9 REPEAT～UNTIL

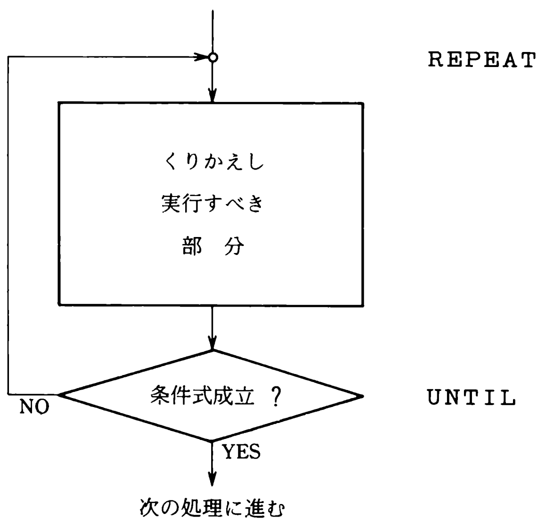
（一部の機種で使用できる．）

指定した条件が満たされるまで反復を行なう場合のために REPEAT，UNTIL という文を使用できる処理系がある*．書き方は，

REPEAT
くりかえし実行すべき部分
UNTIL　条件式

である．

概念図



（例）

```
10 A=1 : B=2
20 REPEAT
30 B=(A+B)/2 : A=2/B
40 PRINT A,B
50 UNTIL ABS(A-B)<1E-05
```

---

* たとえば Hu BASIC

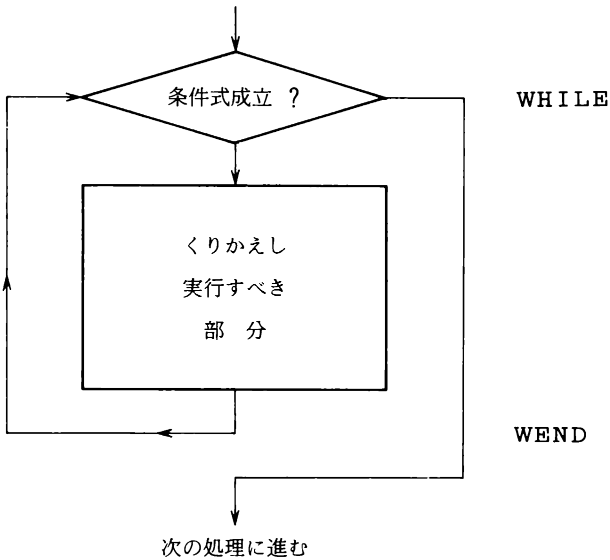
4.10　WHILE～WEND

（これを使用できない機種もある．）

指定した条件が満たされている間だけ反復を行ないたい場合のためにWHILE，WENDという文を使用できる処理系がある*．書き方は

WHILE　条件式
くりかえし実行すべき部分
WEND

概念図



（例）

```
10 A=1 : B=2
20 WHILE ABS(A-B)>1E-05
30 B=(A+B)/2 : A=2/B
40 PRINT A,B
50 WEND
```

---

* たとえばPC-8801のディスクBASIC，Micro-8，*if*-800，Hu BASICなど．

4.11 SWAP

（使用できない機種もある．）

> これは二つの変数の内容を交換する命令で，制御文ではないが，制御文と一緒によく用いられる．SWAP文は次の形で書く．
>
> SWAP　変数名，変数名
>
> (例)　SWAP　A，B

(例)　手もとにパソコンがあったら，次のプログラムを実行させてみるとよい．

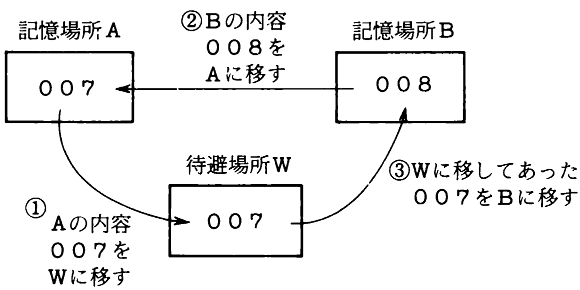
```
10 A=2
20 B=3
30 SWAP A,B
40 PRINT "A=";A
50 PRINT "B=";B
```

内容の入れ替わっているのがわかるであろう．

SWAP文を使えない場合　機種によってはSWAP文を使えないものがある*．そういう機種でSWAP文と同じこと（二つの変数の内容の交換）を行なうには，待避場所を一つ用いて次のように書けばよい．

```
W=A
A=B   } が SWAP A,B と同等
B=W
```

[**動作解説**]　たとえば，Aには7，Bには8が入っているとする．



* たとえばPC-6001．また，MZ-80のSWAP文は本節の意味と全く異なるので注意．

4.12 配　　列

一組のデーターに番号を付けて規則的に並べたものを**配列**（array）という．これは要するに表（ひょう）のことであるが，コンピューターの用語では配列というのが普通である．

データーを一列に並べたものを1次元配列という．

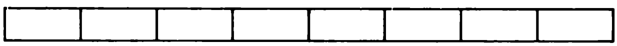
データーを四角に並べたものを2次元配列という．

データーを立体的に並べたものを3次元配列という．

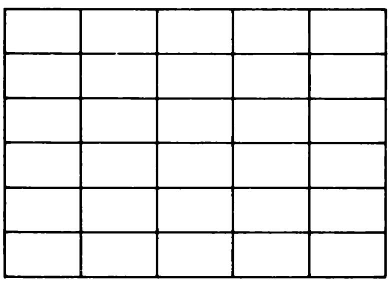
（以下同様に何次元でも可）

配列名　配列には名前を付けて扱う．この名前を配列名という．配列名の付け方（使用文字や長さについての制限など）は変数名の付け方と同じである*．

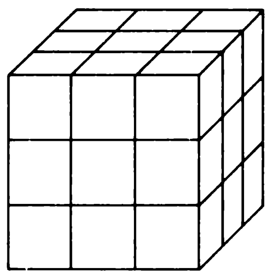
1次元配列



2次元配列



3次元配列



* 基本BASICでは「配列名は英字1字に限る」となっているが，パソコンのBASICにおいては（一部の機種を除けば）変数名と同様，長い配列名を許している．

4.13 添字付き変数

配列に属する個々のデーターのことを**配列要素**という．配列要素は，配列名に添字（いいかえれば番号）を付けた形（これを**添字付き変数**という）で参照することができる．その書き方は

配列名（添字）　　　　　………1次元配列の場合

配列名（第1添字，第2添字）　………2次元配列の場合

（以下同様）である．2次元配列の場合，第1添字は行番号，第2添字は列番号を表す．添字に負数は使用できないが，0は使用してよい．

（例）　配列A

	1列目	2列目	3列目
1行目	1.8	7.3	8.6
2行目	5.2	2.6	4.7
3行目	3.5	0.1	9.8

配列X

1番目	2.9
2番目	0.8
3番目	6.6
4番目	1.5

の場合，A(2,3)の値は4.7，A(3,2)の値は0.1，X(1)の値は2.9，A(1,3)+X(2)の値は8.6+0.8=9.4になる．

添字式　　添字としては，整定数のほか，変数名や式を書いてもよい．

（例）　A(I,J)　　X(2*N−1)

添字として用いる式の形についての制限はなく，どんな式でも書ける．

（例）　C(I/J+SIN(X(N))+5.8)

添字に端数（小数点以下）がある場合は四捨五入により整数化するのが正式であるが，単純切り捨てにしている機種もある．

添字付き変数の引用　　添字付き変数は普通の変数と全く同様に使用することができる．ただしFŌR文の制御変数としては使用できない．

（例）	式の中に書く	A(I)+B
	代入先に指定	A(6)=X*Y+Z
	入　力	INPUT P(3)
	出　力	PRINT Q(7)

4.14 配列の宣言と解除

DIM 文　配列を使用する場合は，あらかじめ，

DIM　配列名（寸法）　…………1次元配列の場合

DIM　配列名（寸法，寸法）　…………2次元配列の場合

……以下同様

などの形で，配列の宣言を行なっておく必要がある．ただし寸法が10以下の場合は宣言せずに配列を用いることができる．

上で寸法と書いたのは添字の上限のことで，コンピューターはその値に合わせて配列の記憶場所を割り当てる．寸法は，あらかじめわかっていればその値を定数として書くのであるが，

(例)　DIM　A(3,3),X(4),C(137)

入力データーや計算の状況に合わせて寸法を決めたい場合には，ここに変数や式を書くこともできる*.

(例)　DIM　A(M,N),X(M+1),C(M+N)

[**解説**]　BASIC の DIM 文は実行文（すなわち，実行時に機能する文）であって，配列の記憶場所の割り当ては，プログラムの流れがそこに到達した時点において行なわれる．したがって，一つのプログラム中に同じ配列の宣言が 2 箇所以上あってもよい（重複して実行されることがなければ）．

ERASE 文［使用できない機種もある］　これは不要になった配列の宣言を解除し，そのぶんの記憶場所を他の目的に転用することができるようにする．これは次のように書く．

ERASE　配列名

(例)　DIM　A(2,3)　で配列Aを2行3列に宣言して使用したあと，不要になったら　ERASE　A　で配列宣言を解除し，別の配列を宣言してもよく，あるいは　DIM　A(5,7)　などと再宣言してもよい．

---

*　ただしコンパイラーでは使用できない．

4.15 配列の入出力

入 力　パソコンの BASIC には一つの命令で配列の全要素の入出力を行なえるような機能はないので，1要素ずつの入出力のくりかえしの形でプログラムを書く必要がある*.

(例)　小さな配列** の入力は次のようにするとよい.

```
10 DIM A(3,2)
20 FŌR I=1 TŌ 3
30     PRINT I;"ぎょうめ を いれてくださ い"
40     INPUT A(I,1),A(I,2)
50 NEXT I
```

この場合，入力データーは1行分ずつ（1行の中ではコンマで区切って）

a_{11}, a_{12}

a_{21}, a_{22}

a_{31}, a_{32}

の順に入れる．一方，大きな配列の入力は次のようにするとよい.

```
10 DIM A(30,50)
20 FŌR I=1 TŌ 30
30     FŌR J=1 TŌ 50
40         PRINT "A(";I;",";J;")=";
50         INPUT A(I,J)
60     NEXT J
70 NEXT I
```

これだと，入力すべきデーターの行番号，列番号が表示されるので，該当するデーターを入れていけばよい.

```
RUN
A( 1 ; 1 )=? 947.2511
A( 1 ; 2 )=? 262.5256
A( 1 ; 3 )=? 233.0641
A( 1 ; 4 )=? 256.3491
A( 1 ; 5 )=? 941.3131
```

---

*　大型コンピューターの BASIC にある MAT READ 文や MAT PRINT文 は，パソコンの BASIC では使用できない.

**　配列の1行分を画面1行に表示できる場合

出 力　出力も同様で，以下の要領でプログラムを書けばよい．

(例) 配列Aが M 行 N 列で，$N \leqq 5$ の場合

```
FŌR I=1 TŌ M          1行目から順に出力
  FŌR J=1 TŌ N
    PRINT A(I,J),     ここでは改行しない
  NEXT J              配列の1行分を1行に出力
PRINT                 改行する
NEXT I
```

［**出力例**］

1	.5	.333333	.25	.2
.5	.333333	.25	.2	.166667
.333333	.25	.2	.166667	.142857
.25	.2	.166667	.142857	.125
.2	.166667	.142857	.125	.111111

(例) 配列Aが M 行 N 列で，$N > 5$ の場合，4列分ずつまとめて出力して，あとで大きな台紙にはり合わせるとよい．

```
FŌR KARA=1 TŌ N-3 STEP 4
  MADE=KARA+3
  IF MADE>N THEN MADE=N
  FŌR J=KARA TŌ MADE
    PRINT "J=";J,
  NEXT J
  FŌR I=1 TŌ M
    FŌR J=KARA TŌ MADE
      PRINT A(I,J),,
    NEXT J
    PRINT
  NEXT I
  PRINT
NEXT KARA
```

4.16 例題および演習

例題 1 —— 2次方程式の根

2次方程式

$$ax^2 + bx + c = 0$$

の根（実根の場合もあり複素根の場合もある）を計算して出力するプログラムを作れ。

［**解答**］ 2次方程式の根の公式を用いる。まず判別式

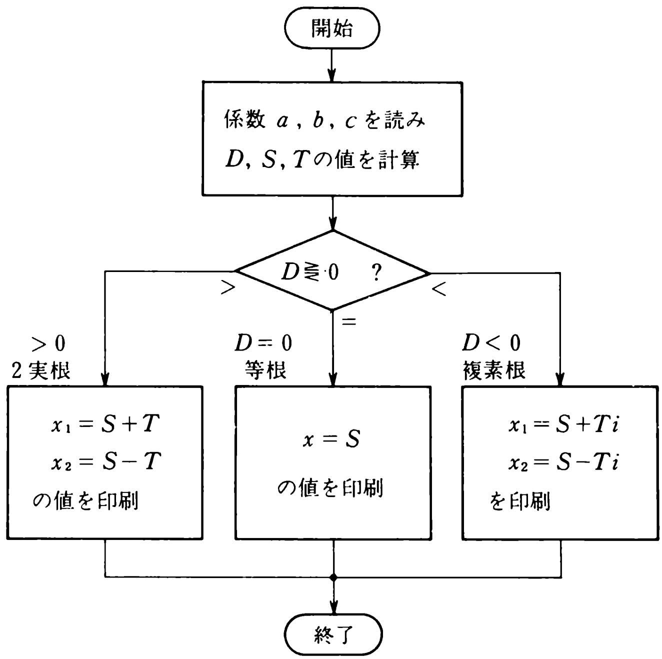
$$D = b^2 - 4ac$$

の値を計算し，ついでに，あとの便宜を考えて

$$S = -b/(2a) \qquad T = \sqrt{|D|}/(2a)$$

の値を計算しておいて，次のように処理する。

［**流れ図**］



[プログラム例]

```
100 REM 2ｼﾞ ﾎｳﾃｲｼｷ a･x^2+b･x+c=0 ｦﾄｸ
110 REM
120 INPUT "a,b,c=";A,B,C
130 REM
140 D=B*B-4*A*C
150 S=-B/(2*A)
160 T=SQR(ABS(D))/(2*A)
170 REM
180 IF D<0 GOTO 300
190 IF D=0 GOTO 260
200 REM
210 REM D>0 ﾉ ﾊﾞｱｲ
220 PRINT "X1=";S+T
230 PRINT "X2=";S-T
240 END
250 REM
260 REM D=0 ﾉ ﾊﾞｱｲ
270 PRINT "X=";S
280 END
290 REM
300 REM D<0 ﾉ ﾊﾞｱｲ
310 PRINT "X1=";S;"+";T;"i"
320 PRINT "X2=";S;"-";T;"i"
330 END
```

[出力例]

```
RUN
a,b,c=? 1,5,6
X1=-2
X2=-3

Ready
RUN
a,b,c=? 25,-10,1
X= .2

Ready
RUN
a,b,c=? 1,0,1
X1= 0 + 1 i
X2= 0 - 1 i

Ready
RUN
a,b,c=? 12.3,45.6,78.9
X1=-1.85366 + 1.72586 i
X2=-1.85366 - 1.72586 i
```

例題 2　数表を作る

1から10までの整数の2乗，逆数，平方根の数表を作成するプログラムを作れ．

［**解答**］ N の値を1から10まで変えて

N　　N^2　　$1/N$　　$\sqrt{N}$

の値を計算し出力すればよい．

```
10 PRINT "N","N^2",                    } 見出しを
20 PRINT "1/N","SQR(N)"                } 出力する
30 FOR N=1 TO 10                       }
40     PRINT N,N^2,1/N,SQR(N)          } 反復
50 NEXT N                              }
```

［**出力例**］

```
N            N^2          1/N          SQR(N)
1            1            1            1
2            4            .5           1.41421
3            9            .333333      1.73205
4            16           .25          2
5            25           .2           2.23607
6            36           .166667      2.44949
7            49           .142857      2.64575
8            64           .125         2.82843
9            81           .111111      3
10           100          .1           3.16228
```

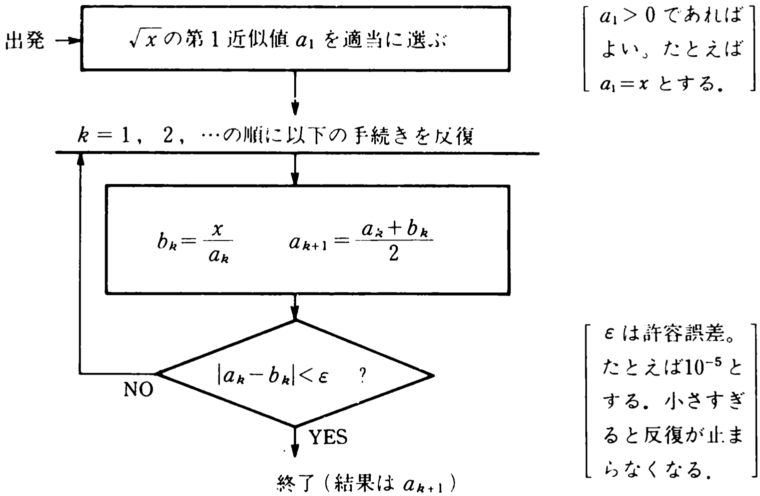
［**備考**］ 1）このプログラムは1行に出力する文字数が多いので，ディスプレイ装置の1行の文字数を60字以上に設定しないと，きれいに出力されない．1行の文字数を多くとれない機種で実行する場合には行10，20，40の出力項目の区切りをセミコロンに変更する必要がある．

2）このプログラムだと，上の出力例のように，整数が左づめになり，右に大きな空白ができる．右づめにして適切な桁数で出力するには次の章で説明する PRINT USING文 を用いる必要がある．

3）プリンターへの出力方法は229ページ参照（比較的簡単である）．

例題 3 ―― 収束判定

$\sqrt{x}$ の値を求めるには組込み関数SQRを用いてもよいが，次のような遂次近似法で計算することもできる．このプログラムを作れ．



[解答]

```
10 EPS=0.000001
20 INPUT "X=";X
30 A=X
40 B=X/A
50 A=(A+B)/2
60 IF ABS((A-B)/A)>EPS GOTO 40
70 PRINT "SQR(";X;")=";A
```

[出力例]

```
RUN
X=? 2
SQR( 2 )= 1.41421

Ready
RUN
X=? 1000000
SQR( 1E+06 )= 1000

Ready
```

問　題

4.1 (大口割り引き) ある店のカセットテープの値段は次のようになっている。

バラ（1本単位）だと，330円（1本につき）

1箱（10本入り）は，3000円（1箱につき）

n 本の値段を計算するプログラムを作れ.

4.2 (妥当性のチェック) 月，日を入力したとき，その値が「あり得ない日付け」(たとえば2月30日とか，13月7日とか，0月−1日など)でないかどうかを調べるプログラムを作れ.

4.3 (うるう年) 西暦年を入力して，その年がうるう年であるかを判定し，その結果を出力するプログラムを作れ.

［ヒント］ 4で割り切れなければ平年. 4で割り切れて，100で割り切れなければうるう年. 100で割り切れて，400で割り切れなければ平年. 400で割り切ればうるう年. なお，N を M で割り切れるかどうかを調べるには，MŌD (34ページ参照) を用いるか，あるいは「N/M」と「$N／M$ の整数部」が等しいか否かを調べればよい. 整数部をとり出すには組込み関数 INT (35ページ参照) を用いる.

4.4 (機能選択) 三角形の面積を計算する公式はいろいろあるが，まず公式番号 n を読み，

$n=1$ ならば，2辺の長さ a，b とそのなす角 θ より

$$S = ab\sin\theta/2$$

$n=2$ ならば，底辺の長さ a と高さ h より

$$S = ah/2$$

$n=3$ ならば，3辺の長さ a，b，c より，Heron の公式

$$S = \sqrt{s(s-a)(s-b)(s-c)} \qquad s = (a+b+c)/2$$

で面積 S を計算するプログラムを作れ.

4.5 （数表を作る） 次の数表を出力するプログラムを作れ．

i） $\theta=0°$から$90°$まで$10°$おきの

$\sin\theta$ $\cos\theta$ $\tan\theta$

の数表．

ii） $x=0.1$ から 2.0 まで 0.1 おきの常用対数 $\log_{10}x$ の数表．

iii） $n=1$ から 10 までの $n!$ の数表．

4.6 （料金早見表） ガソリン・スタンド用の料金早見表を作りたい．単価を4種類（製品の種類などによって異なる．例えば，リッターあたり138円，145円，155円，170円）読み込み，1リッターから0.1リッターおきに30リッターまでの料金表を出力するプログラムを作れ．

4.7 （複利計算） 元金1万円を複利で運用したら1年後，2年後，…，20年後にいくらになるか（元利合計）の表を作りたい．3段階の年利率（例えば，3.5%，5%，6%）を読み込み，上記の表を出力するプログラムを作れ．

4.8 （積立預金） 毎年1万円を積立預金したら，1年後，2年後，…，20年後の元利合計がいくらになるかについて，前問と同様の表を出力するプログラムを作れ．

4.9 （年金） 退職金1000万円を原資とし，毎年100万円ずつ引き出して使用したとして，10年後までの毎年の元利合計を計算するプログラムを作れ．ただし預入れ時点には引き出さず，1年後から引き出すものとする．また，利率は問題4.7と同様に3段階の値を指定できるようにする．

4.10 (極限値) 数値計算によって数列の極限値の厳密な値を求めることはできないが，極限値に近い値をいろいろ計算してみることによって極限値を推測することはできる．たとえば

$$\alpha = \lim_{x \to 0} x^x$$

の値を知るには，$x_n = 10^{-n}$ に対する $\alpha_n = x_n^{x_n}$ の値を $n = 1, 2, 3, \cdots$ に対して計算してみればよいであろう．

i） $n = 1$ から10までに対し，上記 α_n の値を計算して出力するプログラムを作れ．

ii） $\alpha = \lim_{x \to 0} \dfrac{\sin x}{x}$ について同様なプログラムを作れ．

iii） $\alpha = \lim_{x \to 0} (1 + \dfrac{1}{n})^n$ について同様なプログラムを作れ．

iv） 半径1の円に内接する正 n 角形の周の長さ a_n と，外接する正 n 角形の周の長さ b_n は次式で計算できる．

$$a_n = 2n \sin \frac{\pi}{n} \qquad b_n = 2n \tan \frac{\pi}{n}$$

$n \to \infty$ とするときの，これらの値の極限値を知るために，上と同様なプログラムを作れ．

v） $\tan x$ を連分数に展開すると次のようになる．

$$\tan x = \cfrac{x}{1 - \cfrac{x^2}{3 - \cfrac{x^2}{5 - \cfrac{x^2}{7 - \cfrac{x^2}{9 - \text{以下同様}}}}}}$$

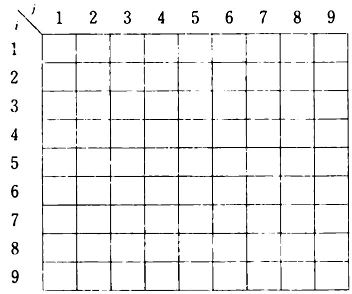
x の値を入力し，上記の式（適切なところまで）で $\tan x$ の値を近似計算するプログラムを作れ．

例題 4 ―― 九九の表

九九の表を作成するプログラムを作れ.

[**解答**] 計算機の内部に右図のような配列を用意し，その i 行 j 列目の所に積 ij の値を書き込んで印刷する.

[**別解**] 1行分だけの配列を用意し，第 i 行の計算をしてすぐ印刷する，という手続きを $i=1$〜9 について反復してもよい.




```
10 DIM A(9,9)              Aが9行9列の配列であることを示す.
20 FŌR I=1 TŌ 9                      iを1から9まで変え，
30     FŌR J=1 TŌ 9                  その中でjを1から9ま
40         A(I,J)=I*J                で変えて，積ijを計算す
50     NEXT J                        る.
60 NEXT I                  これで配列Aの中に九九の表ができた.
70 FŌR I=1 TŌ 9
80     FŌR J=1 TŌ 9
90         PRINT TAB(4*J);A(I,J);
100    NEXT J
110    PRINT               これは「改行せよ」という意味になる.
120 NEXT I
```

[**出力例**]

```
1   2   3   4   5   6   7   8   9
2   4   6   8   10  12  14  16  18
3   6   9   12  15  18  21  24  27
4   8   12  16  20  24  28  32  36
5   10  15  20  25  30  35  40  45
6   12  18  24  30  36  42  48  54
7   14  21  28  35  42  49  56  63
8   16  24  32  40  48  56  64  72
9   18  27  36  45  54  63  72  81
```

問　題

4.11 （素数） 正の整数 N を入力し，N が素数であるか否かを判定するプログラムを作れ．

［ヒント］ 2から $\sqrt{N}$ までのすべての整数で割ってみればよい．N が M で割り切れるかどうかを調べるには，

N MŌD M

が0かどうかを調べればよい．

4.12 （素数表） 1000までの素数を全部求めて出力するプログラムを作れ．

［ヒント］「Eratosthenes のふるい」という方法を用いるとよい．

1000までの素数を求めるのであれば，まず2から1000までの表を用意し，2の倍数全部，3の倍数全部，…に印を付ける（ただし，1倍，すなわち自分自身は除く）．こうして 1000以下のすべての数の倍数に印を付けたとき，残ったものが素数である．

2	3	~~4~~	5	~~6~~	7	~~8~~	~~9~~	~~10~~	…

これをコンピューターで行なうには，寸法1000の配列Aを用意し，その要素全部に0を入れておき，

```
100 FŌR K=2 TŌ SQR(1000)
110     FŌR I=2 TŌ 1000/K
120         L=I*K
130         IF L>1000 THEN 160
140         A(L)=1
150     NEXT I
160 NEXT K
```

というぐあいに処理していけばよい．なお，行100と行110の間で，Kが素数であるかどうかを調べ，素数でなければ行160までスキップするようにすれば，むだが少なくなり，処理時間を大幅に短縮できる．また，行110から始まるループも，よく考えてみると，K=I−1までは既に印がついているのであるから，

```
FŌR I=K TŌ 1000/K
```

としてよい．

4.13 （素因数分解） 正の整数 N を入力し，これを素数の積の形に分解するプログラムを作れ．ただし $N \leqq 100$ とし，100までの素数表を記憶させて利用してよい．

［ヒント］ 下記の要領で処理すればよい．

$K = 2, 3, 5, 7, 11, 13, \cdots$の順に

　$K > N$ ならば終了．

　N を K で割ってみる．割り切れなければ（次の K へ）

　── 割り切れたならば ──

　K を出力．

　新 N ＝現 N / K

　$N > 1$ ならば前にもどって反復

　$N = 1$ ならば終了．

次の K へ

100までの素数は下記の25箇である．

2　3　5　7　11　13　17　19　23　29　31　37　41　43　47　53　59
61　67　71　73　79　83　89　97

出力は，たとえば $N = 24$ の場合

```
2    2    2    3
```

という形にするのが最も簡単であるが，工夫すれば

```
24=2*2*2*3
```

（またはこれに多少，空白が入った形）にすることが可能であろう．

4.14 （素数の分布） 問題4.12（素数表）のプログラムを改造して，

100以下の素数の箇数

101から200までの範囲の素数の箇数

901から1000までの範囲の素数の箇数

を数えて出力するプログラムを作れ．

［ヒント］ 86ページの「区分計算の方法」を参考にするとよい．

例題 5 ―― 階乗，2 項係数

n と r の値を読み，2 項係数（組合せの数）

$$ {}_nC_r = \binom{n}{r} = \frac{n!}{r!(n-r)!} $$

の値を計算して出力するプログラムを作れ．

［**解答**］ 階乗の計算が 3 箇所にあるので，階乗を文200から始まるサブルーチンで計算することにする．

［**プログラム例**］

```
 10 INPUT N,R                     nとrを読む.
 20 K=N                         }
 30 GŌSUB 200                   } n!を計算し，結果を
 40 F1=F                        }   F1に入れる.
 50 K=R                         }
 60 GŌSUB 200                   } r!を計算し，結果を
 70 F2=F                        }   F2に入れる.
 80 K=N-R                       } (n-r)!の計算. 結果
 90 GŌSUB 200                   }   はFに入る.
100 C=F1/(F2*F)
110 PRINT "C=";C
120 STŌP
                                  ------------------------
200 F=1                            ここからサブルーチン.
210 FŌR I=1 TŌ K                  k!の値を計算して結果
220     F=F*I                     をFに入れる.
230 NEXT I
240 RETURN                         サブルーチンの出口.
```

5　基本的なプログラム技法

5.1　整数の計算

関数 INT　整数の計算のコツはINTという関数を活用することである。これは（35ページの表にあるように）

INT(x)＝「xを超えない最大の整数」

という関数である。

(例)　INT(3.18)=3　INT(−3.18)=−4

余　り　正の整数 M を正の整数 N で割った余りを求めるには，次のようにすればよい。

R=M MŌD N

(例)　$M=20$，$N=6$ ならば，$M/N=3.333\cdots$

INT(M/N)=3　R=2

約　数　正の整数 M が正の整数 N で割り切れるかどうかを調べるには

IF M MŌD N=0 THEN …

とすればよい。THENの行先が「割り切れた場合」，分岐せずにIF文の次に進むのが「割り切れなかった場合」である。

整数計算の誤差　整数の計算では，普通は誤差は入らないが，BASICでは整数値も浮動小数点表現で扱っているから，オーバーフロー（overflow，あふれ）は起こさないかわり，仮数部桁数を超えれば誤差が入るので注意を要する。なお，標準規格では「有効桁数6桁以上」となっている。普通の計算は6桁でもよいが，金銭の計算などの際には十分に注意されたい。

5.2　最大公約数と最小公倍数

■　正の整数 M，N の最大公約数 G を求めるには「両者を M（または N）以下のすべての整数で割ってみる」という方法が最も簡単であるが，もっと能率のよいやり方として **Euclid の互除法**というのがある．その手順は次のとおり．

1) M と N を比較し，$M<N$ ならば交換して $M \geqq N$ にする．
2) M を N で割った余り R を求める．
3) 余りが 0 かどうかを調べて，
   - 0 ならば計算終了（最大公約数は N である）
   - 0 でなければ，新 $M=N$，新 $N=R$ として 2)にもどる．

（例）　$M=20$，$N=12$ の場合，

20÷12の余りは 8　　新 $M=12$，新 $N=8$
12÷8 の余りは 4　　新 $M=8$，新 $N=4$
8÷4 の余りは 0 だから計算終了．最大公約数は 4．

■　M，N の最小公倍数 L を求めるには，M と N を掛けて，M と N の最大公約数 G で割ればよい．すなわち，$L=MN/G$.

［プログラム例］

```
10 INPUT MO,NO
20 M=MO
30 N=NO
40 IF M<N THEN SWAP M,N
50 R=M MOD N
60 IF R=0 GOTO 100
70 M=N
80 N=R
90 GOTO 50
100 PRINT "さいだ゛い こうやくすう ";N
110 L=MO*NO/N
120 PRINT "さいしょう こうば゛いすう ";L
```

［出力例］

```
RUN
? 24,36
さいだ゛い こうやくすう 12
さいしょう こうば゛いすう 72
```

5.3 分数の計算，複素数の計算

分数は小数になおして計算するのが最も簡単であるが，分数の形のまま扱うことも可能である．それには，筆算の場合と同様，分母の計算，分子の計算，結果の約分などを，プログラムで書いてやればよい．

［プログラム例］ 分数の加算のプログラムを作ってみよう．公式は

$$\frac{B}{A}+\frac{D}{C}=\frac{BC+AD}{AC}$$

であるから，最も簡単に済ませるつもりなら，次のようにすればよい．

```
10 INPUT A,B,C,D
20 E=A*C
30 F=B*C+A*D
40 PRINT B;"/";A;"+";D;"/";C;
   (続き) "=";F;"/";E
```

もっとていねいにやるとすれば，次のようになる．

1）AとCの最小公倍数Lを求める（前ページ参照）
2）E＝L
3）F=B*(L/A)+D*(L/C)
4）EとFの最大公約数Gを求める（前ページ参照）
5）新E＝E/G，新F＝F/G
6）必要ならば，仮分数を帯分数になおす

複素数の計算

これも同様に，実数部と虚数部それぞれの計算をすればよい．公式は

$$(a+ib)\pm(c+id) = a\pm c+i(b\pm d)$$

$$(a+ib)\cdot(c+id) = (ac-bd)+i(ad+bc)$$

$$\frac{a+ib}{c+id} = \frac{ac+bd}{c^2+d^2}+i\frac{bc-ad}{c^2+d^2}$$

$$|a+ib| = \sqrt{a^2+b^2} \qquad \overline{a+ib} = a-ib$$

5.4 丸　め

正の数の，小数点以下を丸める方法

切り捨て	関数INTを用いればよい．
	Y＝INT（X）
切り上げ	符号を逆にして切り捨てて符号をもとにもどす．
	Y＝−INT（−X）
四捨五入	0.5を加えて切り捨てる．
	Y＝INT（X＋0.5）

負の数の，小数点以下を丸める方法　負数の場合の扱いは，

絶対値について，切り上げ，切り捨て，四捨五入する
符号付きの意味で，切り上げ，切り捨て，四捨五入する

の二通りある．前者ならば，次のようにする．

切り捨て　Y＝−INT（−X）
切り上げ　Y＝INT（X）
四捨五入　Y＝−INT（−X＋0.5）

後者ならば，次のようにする．

切り捨て　Y＝INT（X）
切り上げ　Y＝−INT（−X）
四捨五入　Y＝INT（X＋0.5）

小数点以外の所で丸める方法

適当な係数10^nを掛けて「小数点の所で丸める」という形の問題に変換し上記の処理をしてから10^nで割ってもとにもどす．

（例）　小数点以下第3位を四捨五入するには

Y＝INT（100＊X＋0.5）／100

ただし，2進法や16進法のコンピューターでは，10進法の小数を正確に表現することができないので厳密な意味での切り捨て，切り上げ，四捨五入は不可能である．（例）　0.15は2進法では正確に表現できない．その近似値が0.15より小さければ，四捨五入して0.1になりうる．

5.5 時分秒の計算

> 時，分，秒の単位を併用して時刻を表すことは日常よく行なわれている．その加減算を行なうには
>
> ｛60進法の加減算のプログラムを作る
> 　すべてを秒単位に換算して加減算を行なう
>
> の二つの方法がある．ＩＦ文の練習や n 進法演算の扱いに慣れるためには前者もよいが，実務には後者の方が簡単でよい．

［プログラム例］「h_1 時 m_1 分 s_1 秒から h_2 時 m_2 分 s_2 秒までは何時間何分何秒か？」という計算（時分秒の引き算）をするプログラムを作ってみよう．

```
 10 PRINT "H1,M1,S1=";          }
 20 INPUT   H1,M1,S1            } データーの入力
 30 PRTNT "H2,M2,S2=";          }
 40 INPUT   H2,M2,S2            }
 50     T1=3600*H1+60*M1+S1     } 秒に
 60     T2=3600*H2+60*M2+S2     } 換算
 70     T=T2-T1                        引き算をする
 80     H=INT(T/3600)
 90     R=T-3600*H                     3600で割った余り
100     M=INT(R/60)
110     S=R-60*M                       60で割った余りが秒
120 PRINT H;"ZIKAN";
130 PRINT M;"HUN";
140 PRINT S;"BYŌU"
```

［出力例］

```
H1,M1,S1=? 3,45,11
H2,M2,S2=? 7,20,30
 3 ZIKAN 35 HUN 19 BYOU
```

5.6 日数計算

西暦 y_1 年 m_1 月 d_1 日から，西暦 y_2 年 m_2 月 d_2 日までの日数を計算するには次のようにするとよい．

【定 石】 年，月，日を別個に引き算して，引けなければ上位の値から借りてくる，という方法も考えられるが，月の日数が一定でないため，かなり複雑なプログラムになる．それよりも，一定の基点（たとえば，1600年1月0日）からの日数を計算するプログラムを作り，その日数で引き算を行なう方が簡単である．

まず，1月0日から m 月 d 日までの日数を計算する方法を考えてみるとよい．それには，各月の0日が1月0日から何日目になるかを電卓で計算して表を作り記憶させておくと簡単である．

1月	2月	3月	4月	5月	6月
0	31	59	90	120	151
7月	8月	9月	10月	11月	12月
181	212	243	273	304	334

これは平年の場合である．うるう年には，3月以降，上記の値に1を加える．上記の値に d を加えれば，1月0日から m 月 d 日までの日数N1がわかる．

次に，1600年1月0日から，y 年1月0日までの日数N2を計算する．それにはいろいろな方法が考えられるが，最も簡単かつ明快なのは，ひとまず1年を365日として計算し，それにうるう年のぶんの補正を加える，という方法である．1年を365日として計算すると，N2の概算値は

$$365 \times y$$

となる．これに「1600年から y 年までの，うるう年の回数」を加えればよい．その回数は組込み関数INTを用いて

```
INT((Y-1600)/4)
```

という式で求めることができる．この式は「西暦を 4 で割り切れる年はうるう年である」という規則に基づくものである*．ところで，暦法には

100で割り切れる年は平年とする

ただし400で割り切れる年はうるう年とする

という規則もある*．この補正を行なうには，

INT((Y−1600)/100)を引き，

INT((Y−1600)/400)を加える

とすればよい．以上をまとめると，プログラムは次のようになる．

```
10 REM つきはし゛めまて゛のにっすうのひょうをよみこむ
20 DIM C(12)
30 DATA 0,31,59,90,120,151,181
40 DATA 212,243,273,304,334
50 FOR I=1 TO 12
60 READ C(I)
70 NEXT I
80 REM ここからか゛ ほんは゛ん
90 INPUT "Y1,M1,D1=",Y,M,D
100 GOSUB 200
110 A1=N
120 INPUT "Y2,M2,D2=",Y,M,D
130 GOSUB 200
140 A2=N
150 A=A2-A1
160 PRINT A;"にち"
170 PRINT
180 GOTO 90
190 REM AD1600ねんからのにっすうをけいさんするフ゜ロク゛ラム
200 Z=Y-1600
210 N2=365*Z+INT(Z/4)-INT(Z/100)+INT(Z/400)
220 N1=C(M)+D
230 N=N1+N2
240 IF (Y MOD 4)<>0 GOTO 300
250 IF (Y MOD 400)<>0 GOTO 280
260 IF (Y MOD 100)<>0 GOTO 300
270 REM うるうと゛し の ほせい
280 IF M<=2 GOTO 300
290 N=N+1
300 RETURN
```

---

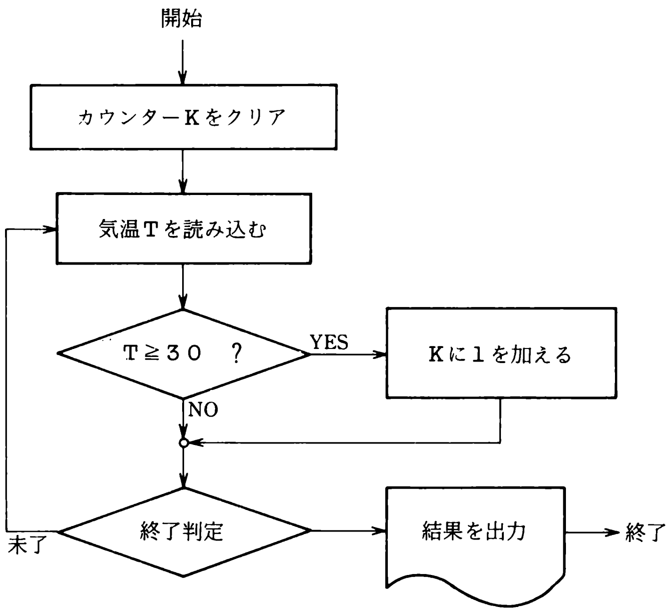
* これは1582年に Gregory 13世が定めた，いわゆるグレゴリオ暦の規則である．1581年以前には適用できない．

5.7 数える

> 箇数を数えるには，箇数を記録するための変数（カウンター）を用意し，最初そこに0を入れておき，順に1を加えていけばよい．複数箇のカウンターを用いれば，分類して数えることもできる．

（例1） 50日間の，正午の気温の観測値 T_1，T_2，…，T_{50} を読み 30℃以上であった日数 K を数えて出力するプログラムは次のようになる．

```
10 K=0
20 FOR I=1 TO 50
30     INPUT T
40     IF T>=30 THEN K=K+1
50 NEXT I
60 PRINT "30どいじょうのひは";K;"にち"
```



（例2） 1000人の学生の都道府県番号（下表参照）N_1, N_2, …, N_{1000} を読み，都道府県別の人数を数えて出力するプログラムを示す．

```
 10 DIM K(47)
 20 FŌR I=1 TŌ 47      ┐ カウンター
 30    K(I)=0          ┘ をリセット
 40 NEXT I
 50 FŌR I=1 TŌ 1000
 60   INPUT N
 70   IF N>47 THEN 140 ────────── (入力の誤り) ─┐
 80   K(N)=K(N)+1                 府県別の計数   │
 90 NEXT I                                        │
100 FŌR I=1 TŌ 40      ┐                          │
110    PRINT I,K(I)    ├ 結果の出力               │
120 NEXT I             ┘                          │
130 END                                           │
140 PRINT "テ゛ーターか゛おかしい" ←──────────────┘
150 END
```

メモ **都道府県コード**（JIS C 6020）

01	北海道	11	埼玉県	21	岐阜県	31	鳥取県	41	佐賀県
02	青森県	12	千葉県	22	静岡県	32	島根県	42	長崎県
03	岩手県	13	東京都	23	愛知県	33	岡山県	43	熊本県
04	宮城県	14	神奈川県	24	三重県	34	広島県	44	大分県
05	秋田県	15	新潟県	25	滋賀県	35	山口県	45	宮崎県
06	出形県	16	富山県	26	京都府	36	徳島県	46	鹿児島県
07	福島県	17	石川県	27	大阪府	37	香川県	47	沖縄県
08	茨城県	18	福井県	28	兵庫県	38	愛媛県		
09	栃木県	19	山梨県	29	奈良県	39	高知県		
10	群馬県	20	長野県	30	和歌山県	40	福岡県		

5.8　合　　計

n箇のデーター a_1, a_2, …, a_n を読み込み,

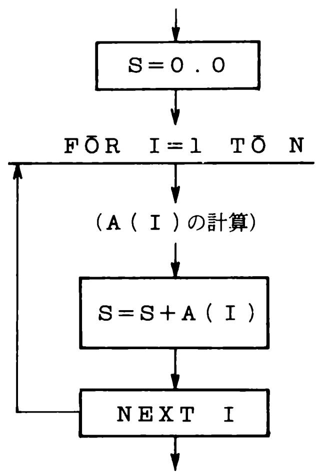
$$S = a_1 + a_2 + \cdots + a_n$$

を計算して印刷するには次のようにする.

> 1）部分和を表す変数を設け,
> その値(出発値)を 0 にしておく.
> 2）第 1 項, 第 2 項, …の順に
> 　↑ 項の値を計算(または読み込む)
> 　│ 項の値を部分和に加える.
> 　└ 最後の項まで反復.
> 3）部分和の最終値が合計値である.

[プログラム例]

```
10  INPUT N
20  DIM A(N)
30  FŌR I=1 TŌ N
50      INPUT A(I)
60  NEXT I
70  S=0
80  FŌR I=1 TŌ N
90      S=A(I)+S
100 NEXT I
110 PRINT "ごうけい ";S
```



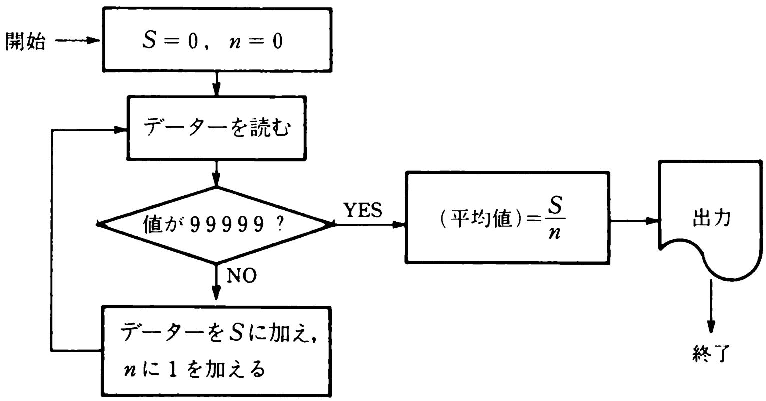
あるいは, 次のようにしてもよい.

```
10  INPUT N
20  S=0
30  FŌR I=1 TŌ N
40      INPUT A
50      S=A+S
60  NEXT I
70  PRINT "ごうけい ";S
```

5.9 不定箇数のデーターの入力

> 一組のデーター a_1, a_2, …, a_n を入力するとき，データーの箇数 n がわかっていれば，n を最初に入力することができるが，場合によっては，n がわからないとか，数えるのがめんどうだから数えないで済ませたいということがある，そのような場合には，最後のデーターに何か印を付けるか，あるいは最後のデーターの次に何か特別な値を入れるかして，入力を終了させればよい．

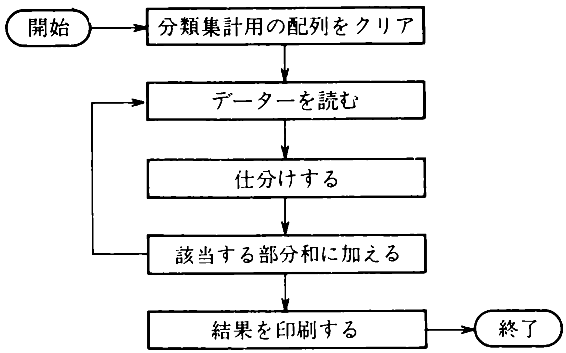
(例) 上記のような方式で，不定箇数のデーターを読み込み，その平均値を計算するには，たとえば最後のデーターの次に99999を入れることにして，次のように処理すればよい．




```
10  S=0  :  N=0
20  INPUT  A
30  IF  A=99999  GŌTŌ  60
40  S=A+S  :  N=N+1
50  GŌTŌ  20
60  PRINT  "へいきんち=";S/N
```

5.10　分類集計

> 合計を求めるには，部分和を記憶するための変数（いわば累計器）を設け，最初そこを0にしておいて，値を順に加算するのであるが，分類集計を行なうには，上記のような「累計器」を分類の数だけ用意し，最初にどれも0を入れておき，各データーを仕分けして該当する「累計器」に加算していけばよい．



（例）　家計簿の整理をするプログラムを作れ．データーとしては

日付　　支出費目コード　　金額

を入れるものとする．日付は年月を省き日のみで，もしも日付が0ならば，「入力データーの終り」と解釈することにする．支出費目コードは，たとえば，食費が1，住居光熱費が2，というように1から5まであるとする．処理としては，日付別の集計と，支出費目別の集計を行なうことにする．

［設計］　日付，支出費目コード，金額を変数名D，C，Aで表し，日付別累計に配列K，費目別累計に配列Lを用いることにする．

配列K

日付	1	2	3	…	…	…	…	31
小計	k_1	k_2	k_3	…	…	…	…	k_{31}

配列L

費目	1	2	3	4	5
小計	l_1	l_2	l_3	l_4	l_5

［プログラム例］

```
 10 DIM K(31),L(5)
 20 FŌR D=1 TŌ 31
 30     K(D)=0
 40 NEXT D
 50 FŌR C=1 TŌ 5
 50     L(C)=0
 70 NEXT C
 80 INPUT D,C,A
 90 IF D=0 GŌTŌ 150
100 IF D>31 GŌTŌ 250
110 IF C>5 GŌTŌ 250
120 K(D)=K(D)+A
130 L(C)=L(C)+A
140 GŌTŌ 80
150 PRINT "ひつ゛けへ゛つこ゛うけい"
160 FŌR D=1 IŌ 31
170     PRINT D,K(D)
180 NEXT D
190 PRINT
200 PRINT "ひもくへ゛つこ゛うけい"
210 FŌR C=1 TŌ 5
220     PRINT C,L(C)
230 NEXT C
240 END
250 PRINT "いれなおしてくた゛さい"
260 GŌTŌ 80
```

20–40: 配列Kをクリア

50–70: 配列Lをクリア

20–70: 省略可

160–180: 日付別分類結果の印刷

210–230: 費目別分類結果の印刷

250–260: 入力ミスなので入れなおしを要求

5.11　平均値と標準偏差

n 箇のデーター x_1, x_2, …, x_n を読み込み，その平均値と標準偏差を計算して印刷するプログラムを作ってみよう．

[計算式]　一般に，n 箇のデーター x_1, x_2, …, x_n の平均値 μ および標準偏差値 σ の計算式は，

$$\mu=\frac{1}{n}\sum_{i=1}^{n}x_i \qquad \sigma=\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i-\mu)^2}$$

[プログラムの設計]　全部のデーター x_1, x_2, …, x_n を記憶装置の中に配列(表)の形で入れて，添字付き変数の形で扱うことにする．手順は

1）データーの格納場所(配列)を用意する．

2）n およびデーター x_1, x_2, …, x_n を読み込む．

3）$\sum x_i$ を計算する．

4）それを n で割る（その結果が μ）．

5）$\sum(x_i-\mu)^2$ を計算する．

6）それを n で割り，その平方根を求める（その結果が σ）．

7）結果を印刷する．

[プログラム例]

```
 10 INPUT "N=";N                                  nを読む
 20 DIM X(N)
 30 FŌR I=1 TŌ N         ┐
 40     INPUT X(I)       ├ データーを読む
 50 NEXT I               ┘
 60 W=0
 70 FŌR I=1 TŌ N         ┐
 80     W=W+X(I)         ├ Σxiの計算
 90 NEXT I               ┘
100 M=W/N                                         平均値μ
```

```
110 W=0
120 FŌR I=1 TŌ N
130     W=W+(X(I)-M)^2
140 NEXT I
150 S=SQR(W/N)
160 PRINT "へいきんち",M
170 PRINT "ひょうし゛ゅんへんさ",S
180 END
```

(120–140: $\sum(x_i-\mu)^2$ の計算)

[別解]

```
 10 N=5
 20 W=0
 30 FŌR I=1 TŌ N
 40     READ X
 50     W=W+X
 60 NEXT I
 70 HEIKIN=W/N
 80 RESTŌRE
 90 W=0
100 FŌR I=1 TŌ N
110     READ X
120     W=W+(X-HEIKIN)^2
130 NEXT I
140 HYŌUJUNHENSA=SQR(W/N)
150 PRINT "へいきんち",HEIKIN
160 PRINT "ひょうし゛ゅんへんさ",HYŌUJUNHENSA
170 DATA 153,177,168,159,161
```

[**備考**] $\sigma=\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2-\mu^2}$ という式は誤差が入るので使わない方がよい.

5.12　相　関　係　数

データー x_1, x_2, …, x_n と y_1, y_2, …, y_n を読み込み，相関係数

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{n\sigma_x\sigma_y}$$

（ただし，$\bar{x}$, $\bar{y}$, σ_x, σ_y は，x_1, x_2, …, x_n および y_1, y_2, …, y_n の平均値および標準偏差）を計算するプログラムを作ってみよう。

```
10 REM テ゛ーターをよみこむ
20 INPUT N
30 DIM X(N),Y(N)
40 FOR I=1 TO N
50 INPUT X(I),Y(I)
60 NEXT I
70 REM X,Yの へいきんちMX,MYをもとめる
80 WX=0
90 WY=0
100 FOR I=1 TO N
110 WX=X(I)+WX
120 WY=Y(I)+WY
130 NEXT I
140 MX=WX/N
150 MY=WY/N
160 REM X,Yの2じ゛ょうわ と せきわ を もとめる
170 SX=0
180 SY=0
190 SXY=0
200 FOR I=1 TO N
210 SX=(X(I)-MX)^2+SX
220 SY=(Y(I)-MY)^2+SY
230 SXY=(X(I)-MX)*(Y(I)-MY)+SXY
240 NEXT I
250 REM ひょうし゛ゅんへんさをSX,SYとする
260 SX=SQR(SX/N)
270 SY=SQR(SY/N)
280 REM そうかんけいすうRをもとめる
290 R=SXY/(N*SX*SY)
300 PRINT R
310 END
```

［**注意**］　下記の公式は誤差が入り易いので使わない方がよい。

$$r = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{\{n\sum x_i^2 - (\sum x_i)^2\}\{n\sum y_i^2 - (\sum y_i)^2\}}}$$

$$= \frac{\sum x_i y_i - n\bar{x}\bar{y}}{\sqrt{\{\sum x_i^2 - n\bar{x}^2\}\{\sum y_i^2 - n\bar{y}^2\}}}$$

5.13 移 動 平 均

データー a_1, a_2, …, a_n の5点移動平均を計算し印刷するプログラムを作ってみよう.

結果の添字の付け方はいろいろあるが，中央点の添字にあわせて

$$b_i = (a_{i-2}+a_{i-1}+a_i+a_{i+1}+a_{i+2})/5 \quad (i=3, \cdots, n-2)$$

で示すことにすればプログラムは次のようになる.

```
10 REM テ゛ーターをよみこむ
20 INPUT "N=";N
30 DIM A(N),B(N)
40 FOR I=1 TO N
50 PRINT "A(";I;")=";
60 INPUT A(I)
70 NEXT I
80 REM いと゛うへいきんのけいさん
90 FOR I=3 TO N-2
100 S=0
110 FOR J=I-2 TO I+2
120 S=A(J)+S
130 NEXT J
140 B(I)=S/5
150 NEXT I
160 REM けっかのしゅつりょく
170 FOR I=1 TO N
180 PRINT A(I),B(I)
190 NEXT I
```

行80〜150を次のようにしてもよい

```
80 REM しゅっは゜つち
85 S=0
90 FOR J=1 TO 5
95 S=A(J)+S
100 NEXT J
105 B(3)=S/5
110 REM くりかえし
120 FOR I=4 TO N-2
130 S=S-A(I-3)+A(I+2)
140 B(I)=S/5
150 NEXT I
```

5.14　行和と列和

データー $x_{ij}(i=1, \cdots, m, j=1, \cdots, n)$ を読み，その

行和　$S_i=\sum_{j=1}^{n} x_{ij}$

列和　$T_j=\sum_{i=1}^{m} x_{ij}$

総計　$Z=\sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij}$

を計算して出力するプログラムを作ってみよう．このような形の計算は会計事務や統計処理によく現われる．

$i \backslash j$	1	2	$\cdots$	n	行和
1	x_{11}	x_{12}	$\cdots$	x_{1n}	S_1
2	x_{21}	x_{22}		x_{2n}	S_2
$\vdots$		$\cdots$			$\vdots$
$\vdots$					$\vdots$
m	x_{m1}	x_{m2}	$\cdots$	x_{mn}	S_m
列和	T_1	T_2	$\cdots$	T_n	Z

［**プログラム例**］　練習であるから m，n は5以下としてよいであろう．

```
 10 DIM X(5,5),S(5),T(5)
 20 PRINT "M,N=";
 30 INPUT M,N
 40 FŌR I=1 TŌ 5
 50  S(I)=0
 60  T(I)=0
 70 NEXT I
 80 Z=0
 90 FŌR I=1 TŌ M
100  FŌR J=1 TŌ N
110   PRINT "X(";I;",";J;")=";
120   INPUT X(I,J)
130   S(I)=S(I)+X(I,J)
140   T(J)=T(J)+X(I,J)
150   Z=Z+X(I,J)
160  NEXT J
170 NEXT I
180 PRINT
```

```
190 FŌR I=1 TŌ M
200  FŌR J=1 TŌ N
210   PRINT X(I,J),
220  NEXT J
230  PRINT "␣I␣";S(I)
240 NEXT I          ↑―――――――――― たて線を引くため
250 FŌR J=1 TŌ N+1
260  PRINT "―――――――――――――――――";
270 NEXT J
280 PRINT "―――"
290 FŌR J=1 TŌ N
300  PRINT T(J),
310 NEXT J
320 PRINT "␣I␣";Z
330 END
```

[**出力例**]

```
 122.6          351.7          I  474.3
 256.8          489            I  745.8
 523.6          872.3          I  1395.9
-------------------------------------------------------
 903            1713           I  2616
```

[**備考**] 上記のプログラムを改造すれば

行ごとの平均値　　列ごとの平均値

行ごとの分散　　　列ごとの分散

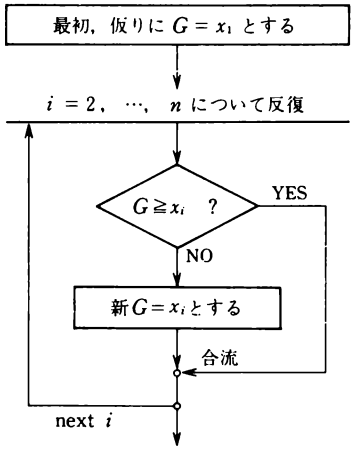
S_1, S_2, …, S_m の分散

T_1, T_2, …, T_n の分散

などを計算できる．このような計算は統計解析の際にしばしば必要になる．

5.15 最大値，最小値

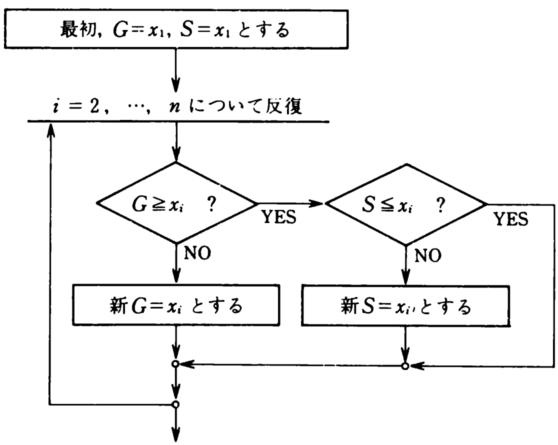
■　一組の値 x_1，x_2，…，x_n の最大値 G を求めるには，次のようにすればよい。



（反復終了時の G の値が最大値である）

■　最小値も同様な要領で求めることができる。

■　最大値 G と最小値 S を同時に求めるには次のようにする。



（反復終了時の G，S の値が最大値，最小値である）

[**プログラム例**] データー a_1, a_2, …, a_n の最大値を求めるには次のようにすればよい。

```
300 MAX=A(1)
310 FŌR I=2 TŌ N
320   IF MAX<A(I) THEN MAX=A(I)
330 NEXT I
```

最大値だけでなく，その番号も必要な場合は次のようにする。

```
300 MAX=A(1) : L=1
310 FŌR I=2 TŌ N
320   IF MAX<A(I) THEN MAX=A(I):L=I
330 NEXT I
```

最小値を求めるには次のようにする

```
340 MIN=A(1)
350 FŌR I=2 TŌ N
360   IF MIN>A(I) THEN MIN=A(I)
370 NEXT I
```

最大値と最小値を同時に求めるには次のようにする。

```
300 MAX=A(1)
310 MIN=A(1)
320 FŌR I=2 TŌ N
330   IF MAX<A(I) THEN MAX=A(I) :
                       (前行の続き) GŌTŌ 350
340   IF MIN>A(I) THEN MIN=A(I)
350 NEXT I
```

絶対値の最大値を求めるには次のようにする。

```
300 MAX=ABS(A(1))
310 FŌR I=2 TŌ N
320   IF MAX<ABS(A(I)) THEN MAX=ABS
                          (前行の続き) (A(I))
330 NEXT I
```

5.16　大きさの順に並べる

一組のデーター x_1, x_2, …, x_n を，大きさの順(大きい順，または小さい順）に並べ替えたいことがよくある．このような処理をソート（**整列**, sorting）という．それは大別して

{ 内部ソート　　主記憶装置の中でソートを行なう
{ 外部ソート　　外部記憶装置を用いてソートを行なう

である．後者は，データーが大量にあって主記憶装置に入りきれない場合に用いられる．内部ソートの方法としては，最大値法(最小値法)，交換法，ヒープ法などがある．外部ソートには，普通，マージ・ソート(merge sort，併合法）という方法が用いられる．

> **最大値法**　大きい順に並べるには，まず最大のものをさがし，次にそれを除いた残りの中で最大のものをさがし，以下同様に「残りの中での最大値」をさがして並べてゆけばよい．

［**プログラム例**］

```
 10 DIM X(100)
 20 INPUT N                    ┐
 30 FŌR I=1 TŌ N               │
 40     INPUT X(I)             ├ データーの読み込み
 50 NEXT I                     ┘
 20 FŌR I=1 TŌ N-1                          第 i 回めには x_i
 70     M=I                                 から x_n までの間
 80     FŌR J=I+1 TŌ N                      の最大値をさがす
 90         IF X(M)<X(J) THEN M=J
100     NEXT J
110     SWAP X(I),X(M)                  今回の最大値X(M)を i
120 NEXT I                              番めの位置にもってくる
130 FŌR I=1 TŌ N               ┐
140     PRINT X(I)             ├ 結果の印刷
150 NEXT                       ┘
```

交換法　これの基本形は「隣り合う二つのデーターを比較し，

正順（望ましい順序）になっていればそのまま

逆順（逆の順序）になっていれば入れ替える

という処理を，列の最初から最後まで何回もくりかえす」という方法である．詳しく検討してみると，反復回数は最大 $n-1$ 回で十分であり，第 i 回めには「列の最初から $n-i$ 番めまで」調べれば十分であることがわかる．

[**プログラム例**]

```
 10 DIM X(100)
 IO INPUT N                  ┐
 30 FŌM I=1 TŌ N             │
 40     INPUT X(I)           │ データーの読み込み
 50 NEXT I                   ┘
 60 FŌR M=N-1 TŌ 1 STEP -1
 70     FŌR J=1 TŌ M
 80        IF X(J)>X(J+1) THEN 120
 90        SWAP X(J),X(J+1)              入れ替え
100     NEXT J
110 NEXT M
120 FŌR I=1 TŌ N             ┐
130     PRINT X(I)           │ 結果の印刷
140 NEXT I                   ┘
```

[**実験**]　上の処理をトランプでやってみるとよい．

1）トランプの中から同じマーク（たとえばハート）のカードだけを集め，よくきって，横に1列に並べる

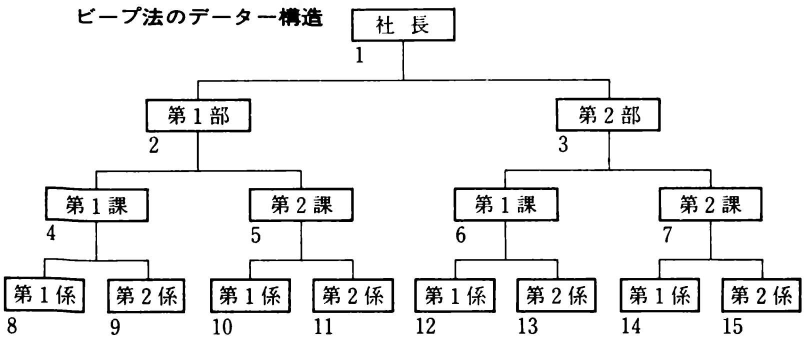
2）左から順に，隣りあう二つを比較し，「左が小，右が大」ならば入れ替える．

3）完全に整列するまで，上記2）を何度でもくりかえす．

5.17　ヒ　ー　プ　法

データーの箇数が100ぐらいまでならば，前節の方法で実用上十分である．しかしデーターの量がもっと多くなると，これらの方法は計算の手間が急増して不利になる．その理由は，要するに，反復第 i 回の処理の際に前回（第 $i-1$ 回）までの情報が十分に生かされないためである．これを改めるため，トーナメント試合の方式を取り入れたのがヒープ法である．

ヒープ法のデーター構造



［**原理**］　上図のような組織の会社を考える．当社では常に

（直系の上司の実力）＞（直系の部下の実力）

という関係が成り立つようにしている．この関係を維持するため，人事異動は次のような方法で行なっている．

1）社長が退任したときは，第1部長と第2部長の実力を比較し，実力のある方を社長に昇格させる．その後任には，その部の第1課長の第2課長の内，実力のある方を昇格させ，その後任には，その課の第1係長と第2係長の内の実力のある方を昇格させる．以下同様．

2）新入社員を採用したときは，一応，最下位に配属するが，ただちにその直属の上司と実力を比較し，逆順なら交代させ，交代した場合はもう一つ上の上司と実力を比較し，逆順ならば交代させる．以下同様．

3）会社設立の際には，社員なしの状態から始めて，上図の番号順にポストをうめてゆき，そのたびに上記2)の処理を行なう．

左記のような組織をヒープ(heap)という．これを用いてソートを行なうには，まずデーターx_1，x_2，…，x_n を上記3)によってヒープの形に並べ，次に「社長」の所から一人ずつ引き抜いて順に並べ，その後任を上記1)の方法でうめてゆけばよい．このような処理をBASICで行なうには左図のような社員番号（配列Xの添字）を用いるとよい．こうすると上司の番号は「部下の番号の1/2の整数部」，また部下の番号は「上司の番号の2倍またはそれプラス1」となり容易に上下のデーターを得ることができる．

［**プログラム**］ ヒープ法のプログラムの一例として，n 組のデーター

x_1 y_1

x_2 y_2

x_n y_n

を読み込み，これを「x の値の小さい順」に並べ替えるプログラム例を以下に示す．実務においては，x（たとえば成績）だけを並べ替えてもあまり意味がなく，それに付随する情報 y（たとえば学生番号）も同時に入れ替えてゆく必要がある．一般には，付随する情報は1箇と限らず，複数箇あるかもしれないが，処理方法は同じなので，ここでは1箇で代表させてある．付随情報の並べ替えのため，最初にJUNという配列に

1 2 3 ……… n

を入れておき，x_i と x_j を入れ替えるときにJUN(i)とJUN(j)も入れ替えている．こうしておけば「最初に i 番の位置にあったデーターが最終的にどこに行ったか」を知ることができ，それを用いて付随するデーターの並べ替えを行なうことができる．じつは，この例のように，付随するデーターが1箇（この例では y）だけの場合は，上記のようにしないで，「x_i と x_j を入れ替えるときに y_i と y_j も入れ替える」という方式にする方が簡単でメモリーも節約になるが，付随する項目数が多い場合は配列JUNを用いる方が能率がよい．

```
10 INPUT"N=";N
20 DIM X(N),Y(N)
30 FOR I=1 TO N
40 INPUT X(I),Y(I)
50 NEXT I
60 REM ***** heap sort *****
70 DIM JUN(N),YQ(N)
80 FOR I=1 TO N
90 JUN(I)=I
100 NEXT I
110 FOR L=2 TO N
120 J=L
130 I=INT(J/2)
140 IF X(I)>=X(J) GOTO 180
150 SWAP X(I),X(J) : SWAP JUN(I),JUN(J)
160 J=I
170 IF J>1 THEN 130
180 NEXT L
190 REM ***** part 2 *****
200 FOR MADE=N TO 2 STEP -1
210 TAIHI=X(1) : NIGEL=JUN(1)
220 I=1
230 J1=2*I
240 J2=2*I+1
250 IF J1=MADE THEN IO=J1 : GOTO 280
260 IF J1>MADE GOTO 310
270 IF X(J1)>X(J2) THEN IO=J1 ELSE IO=J2
280 X(I)=X(IO) : JUN(I)=JUN(IO)
290 I=IO
300 GOTO 230
310 X(IO)=X(MADE) : JUN(IO)=JUN(MADE)
320 I=IO
330 K=INT(I/2)
340 IF X(K)>=X(I) GOTO 380
350 SWAP X(I),X(K) : SWAP JUN(I),JUN(K)
360 I=K
370 IF I>1 THEN 330
380 X(MADE)=TAIHI : JUN(MADE)=NIGEL
390 NEXT MADE
```

```
400 REM ***** Yの ならべ゛かえ *****
410 FOR L=1 TO N
420 YQ(L)=Y(L)
430 NEXT L
440 FOR L=1 TO N
450 LQ=JUN(L)
460 Y(L)=YQ(LQ)
470 NEXT L
480 REM ***** しゅつりょく *****
490 PRINT "ならべ゛かえ の けっか"
500 FOR I=1 TO N
510 PRINT X(I),Y(I)
520 NEXT I
```

［**解説**］ 行10～50はデーターの入力，行80～100は配列ＪＵＮの初期値１，２，…，n の格納である．行110～180がヒープ・ソートの第１部で，「第 L 人めの社員の新規採用とその位置づけ」の処理である．ここで J は実力比較を行なう部下の番号，I はその上司の番号で，最初は「いま入った社員 L」が J となり，その上司と実力比較を行なうが，そこで入れ替えが起こった場合は，その上の階層での比較を順に行なってゆく．こうして第１部が終る時点までに「X の値の大きい順」のヒープができあがる．

第２部（行190～390）では，ヒープから「X の値の大きい順」にデーターをとり出し，それを

$$x_n \quad x_{n-1} \quad x_{n-2} \quad \cdots \quad x_2 \quad x_1$$

の所に順に入れてゆく（だから最終的には「小さい順」になるわけである）．ヒープの上からデーターをとり出して，席があくと，その部下２人（番号Ｊ１とＪ２）の実力を比較し，大きい方を昇格させ，その空席を同様にしてうめてゆく．最下位に達したら行310に進み，最下位の空席に「社員番号の最後」（変数名ＭＡＤＥ）のデーターを移し，それに伴う地位補正を行ない（行330～370），空席ＭＡＤＥの所に先ほど待避させておいた「前社長」を格納する．なお，結果をＸと別の所に入れることにすれば，行320～370の処理は不要になる．

5.18　演　　習

5.1　（複素数の加減乗法）　二つの複素数 $z_1 = a + ib$, $z_2 = c + id$ を読み込み，その和，差，積，商を計算して出力するプログラムを作れ．

5.2　（曜日）　西暦 y 年 m 月 d 日が何曜日かを調べるプログラムを作れ．

［ヒント］　5.6節で説明した方法で「1600年1月0日からの日数」N を計算し，それを7で割った余りを R とすれば，次のようになる．

R	0	1	2	3	4	5	6
曜日	土	日	月	火	水	木	金

5.3　（男女別の平均，相関係数）　身体検査の測定結果を，

性別コード，身長，体重

の形で入力し（性別コードは男が1，女が2である．JIS C-6264）

1）身長の男女別の平均値

2）体重の中央値（メディアン）

3）身長と体重の相関係数

を計算して出力するプログラムを作れ．

5.4　（金種別計算）　給料の支払いは銀行振込みの会社が多くなったが，まだ現金で支給している所も少なくない．現金で渡す場合には，各人の給料袋に分けて入れる関係上，1万円札のほかに，5千円札，千円札，500円玉，百円玉，十円玉を，必要な枚数だけ用意しなければならない．その必要枚数を計算するのが金種別計算である．従業員 n 人($n \leqq 100$とする)の今月の現金支給額 a_1, a_2, …, a_n を読み込み，金種別計算を行なうプログラムを作れ．

［ヒント］　各個人の必要枚数は次のようにして計算できる．

（現金支給額）÷（1万）＝（1万円札枚数）余り（1万円未満）

（1万円未満）÷（5千）＝（5千円札枚数）余り（5千円未満）

（5千円未満）÷（　千）＝（　千円札枚数）余り（　千円未満）

以下同様

6* 数 値 計 算

6.1 パソコンで解ける問題の例

パソコンでも，本格的な数値計算が可能である．参考までに，「こんな問題が簡単に解ける」という二，三の例を紹介する．

連立1次方程式　本章6.15節のプログラムで50元ぐらいまでならば解ける．所要時間は

30元で約3分半

50元で約15分

程度である（PC-8001による実測値）．

高次代数方程式　「高次」といっても，むやみに次数の高いもの（たとえば100次方程式）は誤差が入り易く，そのため大型コンピューターでも簡単には解けない．しかし10次ぐらいまでならば，本章6.10節のプログラムでたいてい解ける．所要時間は（収束のぐあいに関係するので問題ごとに異なるが）たとえば7次方程式

$$x^7+x^6+x^5+x^4+x^3+x^2+x=0$$

の場合，約15秒である（Level-3による実測値）．

常微分方程式　連立，高階，あるいは非線形であっても比較的容易に解けるのが数値解法の強みである．

（例）問　題　非線形振動

方程式　$y''+\sin y=0$　初期条件　$y(0)=3,\ y'(0)=0$

計算範囲　$0 \leqq x \leqq 10$　進み幅　$\varDelta x=0.1$

所要時間　48秒

＊　数値計算に興味がなければ，とばして次の章に進んでよい．

6.2 利用者が定義する関数

組込み関数（3.7節参照）以外の関数は，自分で作らなければならない．それを一つのプログラムの中の何箇所もで使うのであれば，サブルーチンの形で扱うのがよいであろう．しかし，特に，関数の計算式が1行で書ける場合には，「利用者が定義する関数」という形で扱うと，

　　引数を設けることができる

　　算術式の中に書ける

という利点がある．

■ 関数を定義するにはDEF文を用いる．これは次の形式で書く．

　　DEF　関数名（仮引数）＝その関数の計算式

■ 関数名は3文字以上で，

　　先頭2文字はFNに限る

　　第3字は英字に限る

（正しい例）　FNA　FNC

（誤りの例）　TAN　F　FUNCTIŌN

■ 仮引数が不要ならば無くてもよい．その場合の形は

　　DEF　関数名＝その関数の定義式

となる．

■ 仮引数としては変数名を書く．本文の変数名と重複可．

■ 関数は，関数名（実引数）の形で代入文などの式の中に書くことができる．実引数としては，定数，変数のほか，式を書いてもよい．引数なしの関数は関数名だけで引用される．

◆ DEF文は，そこをプログラムの流れが通過するときに関数が「定義」される．単に「プログラムの一部に書いてある」だけでは有効にならないことがあるので注意を要する．

6.3 自分で簡単に作れる関数

下記の関数は組込み関数に入っていないが，DFE文によって容易に自作することができる．

$$\sinh x = \frac{e^x - e^{-x}}{2}$$

$$\cosh x = \frac{e^x + e^{-x}}{2}$$

$$\tanh x = \sinh x / \cosh x$$

$$\arcsin x = \arctan \frac{x}{\sqrt{1-x^2}}$$

$$\arccos x = \pi/2 - \arcsin x$$

$$\sqrt[3]{x} = x^{1/3}$$

［**プログラム例**］

上記の関数を BASIC で書いた例を示す．

```
DEF FNSH(X)=(EXP(X)-EXP(-X))/2
DEF FNCH(X)=(EXP(X)+EXP(-X))/2
DEF FNTH(X)=FNSH(X)/FNCH(X)
DEF FNAS(X)=ATN(X/SQR(1-X*X))
DEF FNAC(X)=1.570796-FNAS(X)
```

［**使用例**］ 上の関数を用いて数表を作成する例を示す．

```
10 DEF FNSH(X)=(EXP(X)-EXP(-X))/2
20 FOR X=0 TO 2 STEP 0.1
30     Y=FNSH(X)
40     PRINT X,Y
50 NEXT X
```

［**出力例**］

```
0           0
.1          .100167
.2          .201336
.3          .30452
.4          .410752
.5          .521095
.6          .636654
.7          .758584
.8          .888106
.9          1.02652
```

6.4 近似式

コンピューターで関数値を計算する方法としては

近似式で計算する

数表を補間して用いる

の二つが基本的である．近似式としては，**Taylor 級数**

(例)

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \cdots + \frac{x^n}{n!} + \cdots$$

$$\sin x = \frac{x}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots + (-1)^k \frac{x^{2k+1}}{(2k+1)!} + \cdots$$

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \cdots + (-1)^k \frac{x^{2k}}{(2k)!} + \cdots$$

を有限項で打ち切って計算することがまず考えられるが，上記の三つの級数は収束が非常に速いので，たとえば $|x| \leqq 1$ の範囲で用いる場合，e^x で10項，$\sin x$，$\cos x$ で5項もとれば誤差を 10^{-6} 以下にすることができる．しかし，収束が非常に遅い級数もある．

(例)

$$\arctan x = x - \frac{x^3}{3} + \frac{x^5}{5} - \cdots + (-1)^k \frac{x^{2k+1}}{2k+1} + \cdots$$

$$\log(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \cdots + (-1)^{k+1} \frac{x^k}{k} + \cdots$$

よく使われる関数については，もっと能率のよい計算法が研究され，近似公式が作られている．

常用対数の簡単な近似式の一例（Hastings の公式）

$$\log_{10} x = c_1\left(\frac{x-1}{x+1}\right) + c_3\left(\frac{x-1}{x+1}\right)^3$$

$$c_1 = 0.86304 \qquad c_3 = 0.3645$$

適用範囲 $\frac{1}{\sqrt{10}} \leqq x \leqq \sqrt{10}$　　最大誤差　約 5×10^{-4}

tan x の近似式の一例（永坂の公式）

$$\tan\frac{\pi}{4}x \cong \left(0.13451446+\frac{2.6252949}{4.0333293-x^2}\right)x$$

適用範囲　$-1\leqq x\leqq 1$　　最大誤差　2.5×10^{-6}

誤差関数の近似式の一例（Hastings の公式）

$$\frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt \cong 1-\frac{1}{(1+a_1x+a_2x^2+a_3x^3+a_4x^4)^4}$$

$a_1 = 0.278393$　　$a_3 = 0.000972$

$a_2 = 0.230389$　　$a_4 = 0.078108$

適用範囲　$0\leqq x<\infty$　　最大誤差　5×10^{-4}

Bessel 関数の近似式の一例（Allen の公式）

$$J_0(x) \cong a_0+a_1\left(\frac{x}{3}\right)^2+a_2\left(\frac{x}{3}\right)^4+\cdots+a_6\left(\frac{x}{3}\right)^{12}$$

$a_0 = 1$　　$a_1 = -2.2499997$　　$a_2 = 1.2656208$

$a_3 = -0.3163866$　　$a_4 = 0.0444479$

$a_5 = -0.0039444$　　$a_6 = 0.0002100$

適用範囲　$-5\leqq x\leqq 3$　　最大誤差　10^{-7}

ガンマ関数　　ガンマ関数 $\Gamma(x)$ の値を計算するには，まず，公式

$$\Gamma(x+1) = x\Gamma(x)$$

をくりかえし使って引数を $1\leqq x\leqq 2$ の範囲に移し，Hastings の近似式(31番)

$$\Gamma(1+x) = 1+a_1x+a_2x^2+a_3x^3+a_4x^4+a_5x^5$$

$a_1 = -0.5748646$　　$a_2 = 0.9512363$

$a_3 = -0.6998588$　　$a_4 = 0.4245549$

$a_5 = -0.1010678$

を用いるのが比較的簡単である．

6.5 級数の和

級数

$$1+\frac{1}{1!}+\frac{1}{2!}+\frac{1}{3!}+\cdots+\frac{1}{10!}$$

の和を計算するプログラムを作ってみよう．

[**設計**]　k の値を 0 から10まで変えて，第 k 項 $a_k = 1/k!$ の値を計算し，合計すればよいであろう．a_k の値を計算するには，$k!$ の値を求めて逆数をとる，という方法も考えられるが，

$$a_k = \frac{1}{k!} = \frac{1}{k}\cdot\frac{1}{(k-1)!} = \frac{1}{k}\cdot a_{k-1}$$

という関係を用いて，

$$a_0 = 1$$
$$a_k = a_{k-1}/k \qquad (k = 1,\ 2,\ \cdots,\ 10)$$

の形で計算する方が簡単である．その合計を求めるには，第 k 項までの部分和を S_k として，やはり漸化式の形

$$S_0 = 1$$
$$S_k = S_{k-1}+a_k \qquad (k = 1,\ 2,\ \cdots,\ 10)$$

で計算することができる．なお，a_k，S_k が計算されれば，以後は，a_{k-1}，S_{k-1} は必要ないから，それを消して同じ場所に書き込むことにすると，記憶場所は一般項の部分和のために各 1 語あればよい．

[**プログラム例**]

```
10 A=1                  一般項，最初は 1
20 S=1                  部分和，最初は 1
30 FŌR K=1 TŌ 10
40     A=A/K            aₖ の計算
50     S=S+A            部分和に加え込む
60 NEXT K
70 PRINT S              結果の印刷
```

6.6　多項式の値の計算法

多項式の値を計算するには，

$$P(x) = a_0x^n + a_1x^{n-1} + a_2x^{n-2} + \cdots + a_{n-2}x^2 + a_{n-1}x + a_n$$
$$= ((\cdots((a_0x + a_1)x + a_2)x + \cdots + a_{n-2})x + a_{n-1})x + a_n$$

と変形し，次の形で計算するとよい．

$S = a_0$

$i = 1,\ 2,\ \cdots,\ n$ の順に
└─ 新 $S = S \cdot x + a_i$

（例）　x の値を読み込み

$$y = 2x^3 + 7x^2 + 6x + 11$$

の値を計算して印刷するプログラムを作ってみよう．

［**プログラム例**］　上記の注意に従い

$$y = ((2x + 7)x + 6)x + 11$$

という形に変形して計算する．

```
10 INPUT X
20 Y=((2*X+7)*X+6)*X+11
30 PRINT Y
```

（例）　係数 a_0，a_1，…，a_n が配列A（添字0，1，…，n）に入っている場合には次のようにする．

```
100 INPUT X
110 Y=A(0)
120 FŌR I=1 TŌ N
130     Y=Y*X+A(I)
140 NEXT I
150 PRINT Y
```

（上記のプログラムをテストするためのデータ入力）

```
10 READ N
20 FŌR I=0 TŌ N
30     READ A(I)
40 NEXT I
50 DATA 3,2,7,6,11
```

6.7 多項式の積

多項式

$$P(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1} + a_n x^n$$

$$Q(x) = b_0 + b_1 x + b_2 x^2 + \cdots + b_{m-1} x^{m-1} + b_m x^m$$

の積

$$R(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_{m+n} x^{m+n}$$

を計算するには，次のようにするのが簡単である．

最初，$c_0 = c_1 = \cdots = c_{m+n} = 0$ にしておく．

$i = 0,\ 1,\ \cdots,\ n$ について

　$j = 0,\ 1,\ \cdots,\ m$ について

　　c_{i+j} に $a_i b_j$ を加える

［プログラム例］

```
 10 INPUT N : DIM A(N)
 20 FOR I=0 TO N
 30     INPUT A(I)
 40 NEXT I
 50 INPUT M : DIM B(M)
 60 FOR I=0 TO M
 70     INPUT B(I)
 80 NEXT I
 90 DIM C(M+N)
100 FOR I=0 TO N
110     FOR J=0 TO M
120         C(I+J)=A(I)*B(J)+C(I+J)
130     NEXT J
140 NEXT I
150 FOR I=0 TO M+N
160     PRINT "C(";I;")=";C(I)
170 NEXT I
```

6.8 組 立 除 法

> n 次多項式
>
> $$P(x)=a_0x^n+a_1x^{n-1}+a_2x^{n-2}+\cdots+a_{n-1}x+a_n$$
>
> を $x-c$ で整除した商
>
> $$Q(x)=b_0x^{n-1}+b_1x^{n-2}+b_2x^{n-3}+\cdots+b_{n-2}x+b_{n-1}$$
>
> と余り R を求めるには，組立除法を用いるとよい．計算式は
>
> $b_0=a_0$
>
> $i=1,\ 2,\ \cdots,\ n$ の順に
>
> └── $b_i=a_i+cb_{i-1}$
>
> $R=b_n$

［**解説**］ 多項式の割り算の要領は普通の数の割り算と同じ．

(例)

$$
\begin{array}{r|l}
 & x+8 \\ \hline
x-3 & x^2+5x-11 \\
 & \underline{x^2-3x} \\
 & 8x-11 \\
 & \underline{8x-24} \\
 & 13
\end{array}
\qquad
\begin{array}{r|l}
 & x^{n-1}+b_1x^{n-2}+b_2x^{n-3} \\ \hline
x-c & x^n+a_1x^{n-1}+a_2x^{n-2}+a_3x^{n-3}+\cdots \\
 & \underline{x^n-cx^{n-1}} \\
 & (a_1+c)x^{n-1}+a_2x^{n-2} \\
 & \underline{b_1x^{n-1}-b_1cx^{n-2}} \\
 & (a_2+b_1c)x^{n-2}+a_3x^{n-3}
\end{array}
$$

その計算式の数値的部分だけを抜き書きしたのが上の公式である．

［**プログラム例**］

```
 10 INPUT "N,C=";N,C
 20 DIM A(N),B(N)
 30 FOR I=0 TO N
 40    INPUT A(I)
 50 NEXT I
 60 B(0)=A(0)
 70 FOR I=1 TO N
 80    B(I)=B(I-1)*C+A(I)
 90 NEXT I
100 FOR I=0 TO N-1
110    PRINT "B(";I;")=";B(I)
120 NEXT I
130 PRINT "R=";B(N)
```

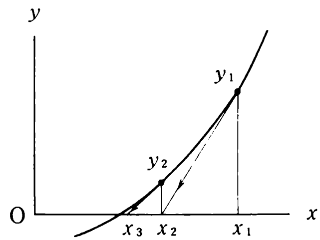
6.9 方程式 $f(x)=0$ の根

方程式 $f(x)=0$ の実根の計算によく使われる方法としては次のようなものがある．

Newton 法　適当な第 1 近似値 x_1 から出発して，

$$x_{k+1}=x_k-\frac{f(x_k)}{f'(x_k)}$$

$(k=1,\ 2,\ \cdots)$ によって精度を高めてゆく．上式は，点 $(x_k,\ y_k)$ において，曲線 $y=f(x)$ の接線を引き，x 軸との交点を x_{k+1} とすることを意味し，要するに $f(x)$ を局部的に 1 次式で近似して近似解を求めているわけである．根に近づくにしたがって，この近似の精度が良くなるため，収束はしだいに速くなる．適当な第 1 近似値がわかっている場合には非常に良い(速い)方法である．



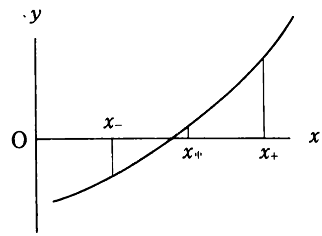
［**プログラム例**］　方程式 $x-\cos x=0$ を解くプログラムを作ってみよう．$f(x)=x-\cos x$，$f'(x)=1+\sin x$ であるから，

$$x_{k+1}=x_k-\frac{x_k-\cos x_k}{1+\sin x_k}$$

を反復すればよい．出発値は $x_1=1$ にとることにする．

```
10 X=1
20 FŌR K=1 TŌ 10
30     F=X-CŌS(X)
40     D=1+SIN(X)
50     X=X-F/D
60     IF ABS(F)<0.00001 GŌTŌ 80
70 NEXT K
80 PRINT "X=";X
```

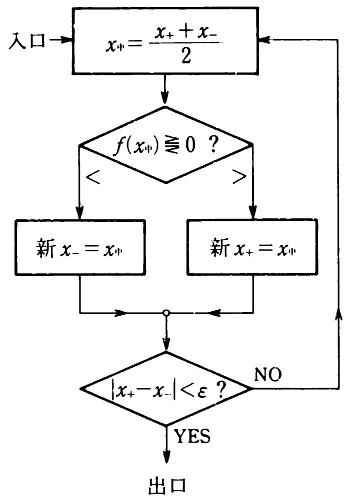
2分法　最初に，$f(x)>0$になる点x_+と，$f(x)<0$になる点x_-をさがしておく．次にx_-とx_+の中点xにおいて関数値を計算し，その符号を調べる．もしこれが正ならば根はx_-と$x_{中}$の間にあり，負ならば根はx_+と$x_{中}$の間にあるから，それによってx_+またはx_-を$x_{中}$で置き換えて同様な計算を反復すればよい．



[プログラム例]　左記の例と同じ方程式$x-\cos x=0$の根を計算する場合のプログラム例を示す．出発値は

$$x_-=0 \qquad x_+=1$$

とする．根の存在範囲は反復1回ごとに1/2になってゆくから，10回反復すれば$|x_+-x_-|=2^{-10}\fallingdotseq 10^{-3}$になる．




```
10  XM=0              (x_)
20  XP=1              (x+)
30  FŌR  K=1  TŌ  10
40       X=(XM+XP)/2
50       F=X-CŌS(X)
60       IF  F<0  THEN  XM=X  ELSE  XP=X
70  NEXT  K
80  PRINT  "X=";X
```

6.10 高次代数方程式

本節では，実係数の n 次代数方程式

$$x^n+a_1x^{n-1}+a_2x^{n-2}+\cdots+a_{n-1}x+a_n=0$$

のすべての根(実根，複素根とも)を計算するための Bairstow-Hitchcock 法について説明する．

[**原理**] 上記の方程式の左辺を $f(x)$ とする．これを，実係数の 2 次式 x^2+px+q と，$n-2$ 次式

$$g(x)=x^{n-2}+b_1x^{n-3}+b_2x^{n-4}+\cdots+b_{n-3}x+b_{n-2}$$

の積に因数分解することを考える*．

$$f(x)=(x^2+px+q)\,g(x)$$

この分解ができれば，2 次方程式 $x^2+px+q=0$ を解いて二つの根が得られ，$g(x)$の方はまた今と同様な因数分解をしてゆくことができるであろう．問題は，いかにして上記のような p，q を求めるかであるが，Bairstow-Hitchcock 法では次のようにする．

まず，p と q の出発値（第 1 近似値）を適当に選ぶ．そして，2 次式 x^2+px+q で $f(x)$を割ってみる．その商を $\hat{g}(x)$，その係数を b_1，b_2，…，b_{n-2}，余りを $rx+s$ とする．割り算の要領は

$$
\begin{array}{r l}
 & x^{n-2}+b_1x^{n-1}+\cdots+b_{n-2} \\
x^2+px+q & \big)\overline{x^n+a_1x^{n-1}+a_2x^{n-2}+\cdots+a_{n-1}x+a_n} \\
 & \underline{x^n+px^{n-1}+qx^{n-2}} \\
 & (a_1-p)\,x^{n-1}+(a_2-q)\,x^{n-2}+a_3x^{n-3} \\
 & \underline{b_1x^{n-1}+b_1px^{n-2}+b_1qx^{n-3}} \\
 & \underline{\text{以下同様}} \\
 & \text{余り}\quad rx+s
\end{array}
$$

計算式は右記1)のとおりで，$r=b_{n-1}$，$s=b_n+pb_{n-1}$ となる．

---

* 実係数の代数方程式の複素根は必ず共役複素数になるので，このような実係数の因数分解が可能である．1 次式と $n-1$ 次式の積に分解しようとすると，一般に複素係数の多項式になり計算に手間がかかる．

p と q を与えると r と s が決まるのであるから，r と s は p と q の関数である．それが同時に 0 になるように，p，q の値を修正する．修正量を Δp，Δq とし 2 変数の Taylor 展開の 1 次の項までとるとすれば

$$\begin{cases} r(p+\Delta p, q+\Delta q) \fallingdotseq r(p,q) + \dfrac{\partial r}{\partial p}\Delta p + \dfrac{\partial r}{\partial q}\Delta q = 0 \\ s(p+\Delta p, q+\Delta q) \fallingdotseq s(p,q) + \dfrac{\partial s}{\partial p}\Delta p + \dfrac{\partial s}{\partial q}\Delta q = 0 \end{cases}$$

を解いて $\Delta p, \Delta q$ を決めればよい．その係数は次のようにして求める．

$$(x^2+px+q)\bar{g}(x) + (rx+s) = f(x) \qquad \text{（定義より）}$$

$$\bar{g}(x) + (x^2+px+q)\frac{\partial g}{\partial q} + \frac{\partial r}{\partial q}x + \frac{\partial s}{\partial q} = 0 \quad \left(\begin{array}{l}\text{上式を } q \\ \text{で偏微分}\end{array}\right)$$

したがって，$g(x) \div (x^2+px+q)$ の余りの，x の係数が $-\partial r/\partial q$ で定数項が $-\partial s/\partial q$．同様に $xg(x) \div (x^2+px+q)$ の余りの x の係数が $-\partial r/\partial p$，定数項が $-\partial s/\partial p$ である．

［**計算手順**］ 以上をまとめると，次のようになる．

（$|r|$，$|s|$ が十分小さくなるまで反復．）

1） $b_1 = a_1 - p \quad b_2 = a_2 - pb_1 - q$

　$i = 3, \cdots, n$ の順に
　└ $b_i = a_i - pb_{i-1} - qb_{i-2}$

2） $c_1 = b_1 - p$，$c_2 = b_2 - pc_1 - q$

　$i = 3, \cdots, n-1$ の順に
　└ $c_i = b_i - pc_{i-1} - qc_{i-2}$

3） 連立 1 次方程式

$$\begin{cases} c_{n-2}\Delta p + c_{n-3}\Delta q = b_{n-1} \\ (c_{n-1} - b_{n-1})\Delta p + c_{n-2}\Delta q = b_n \end{cases}$$

を解いて Δp，Δq を求める．

4） 新 $p = p + \Delta p$，新 $q = q + \Delta q$

5） $x^2 + px + q = 0$（その根がもとの方程式の根になる）

新 $a_i = b_i (i = 1, 2, \cdots, n-2)$，新 $n = n-2$ として，手順 1 にもどる．ただし $n \leqq 2$ ならば直接解いて計算終了にする．

```
10 REM ##########################
20 REM # こうじ゛ だ゛いすうほうていしきを とく #
30 REM #       Bairstowほう       #
40 REM ##########################
50 REM
60 REM ------- デ゛ーターのよみこみ -------
70 INPUT "N=";N
80 DIM A(N),B(N),C(N)
90 DIM X(N),Y(N)
100 FOR I=1 TO N
110 PRINT "Xの";N-I;"じ゛ょうのけいすう";
120 INPUT A(I)
130 NEXT I
140 REM しゅうそく はんていの じ゛ょうすう
150 E=0.001
160 REM さいだ゛い はんぷ゜く かいすう
170 KMAX=100
180 REM ---- こんを 2こず゛つ もとめる ----
190 FOR M=N TO 3 STEP -2
200 FOR KURIKAESI=1 TO 5
210 REM PとQの しゅっぱ゜つちを らんすうで゛ きめる
220 P=RND
230 Q=RND
240 REM ------- メイン・ループ゜ -------
250 FOR K=1 TO KMAX
260 REM ... Bのけいさん ...
270 B(1)=A(1)-P
280 B(2)=A(2)-P*B(1)-Q
290 FOR I=3 TO M
300 B(I)=A(I)-P*B(I-1)-Q*B(I-2)
310 NEXT I
320 REM ... Cのけいさん ...
330 C(1)=B(1)-P
340 C(2)=B(2)-P*C(1)-Q
350 IF M>3 GOTO 380
360 A2=1
370 GOTO 430
380 FOR I=3 TO M-1
390 C(I)=B(I)-P*C(I-1)-Q*C(I-2)
400 NEXT I
410 REM ... ほせいりょうのけいさん ...
420 A2=C(M-3)
430 A1=C(M-2)
440 A3=C(M-1)-B(M-1)
```

```
450 A4=C(M-2)
460 B1=B(M-1)
470 B2=B(M)
480 D=A1*A4-A2*A3
490 P1=(B1*A4-B2*A2)/D
500 Q1=(A1*B2-A3*B1)/D
510 REM ... PとQのほせい ...
520 P=P+P1
530 Q=Q+Q1
540 REM ... しゅうそく はんてい ...
550 IF ABS(B1)+ABS(B2)<E GOTO 600
560 NEXT K
570 NEXT KURIKAESI
580 PRINT "はんぷく うちきり"
590 STOP
600 D=P*P-4*Q
610 IF D<0 THEN 690
620 REM ------ じっこんの ばあい -------
630 X1=(-P+SQR(D))/2
640 X2=(-P-SQR(D))/2
650 PRINT X1
660 PRINT X2
670 GOTO 740
680 REM ------ ふくそこんの ばあい -------
690 R=-P/2
700 S=SQR(-D)/2
710 PRINT R;"+";S;"I"
720 PRINT R;"-";S;"I"
730 REM -------- デフレーション --------
740 IF M<=2 GOTO 870
750 FOR I=1 TO M-2
760 A(I)=B(I)
770 NEXT I
780 NEXT M
790 REM --- のこり の こんを しゅつりょく ---
800 IF N=INT(N/2)*2 GOTO 830
810 PRINT -B(1)
820 GOTO 870
830 P=B(1)
840 Q=B(2)
850 M=2
860 GOTO 600
870 PRINT "おわり"
880 END
```

6.11 数表の使い方

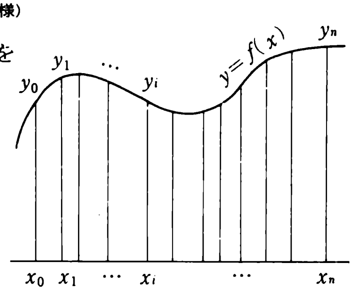
表の読み込み　数表はDATA文で書いておいてREAD文で読み込むとよい。

［プログラム例］

```
10 READ N
20 DIM X(N),Y(N)
30 FŌR I=1 TŌ N
40     READ X(I),Y(I)
50 NEXT I
60 DATA n の値
70 DATA x1 の値, y1 の値
80 DATA x2 の値, y2 の値
                        ……（以下同様）
```

表の引き方　関数 $y=f(x)$ の数表を

x_0	y_0
x_1	y_1
x_2	y_2
⋮	⋮
x_n	y_n



とする。引数（独立変数の値）が与えられたとき，$x_l \leqq x < x_{l+1}$ になるような l をさがすには次のようにすればよい。

分点 x_i が等間隔の場合　分点間隔を h とすれば（すなわち $x_i=x_{i-1}+h$　$(i=1,\ 2,\ \cdots,\ n)$　とすれば），l は次のようにして求められる。

$$l=「(x-x_0)/h \text{ の整数部分}」$$

BASIC で書けば次のようになる。

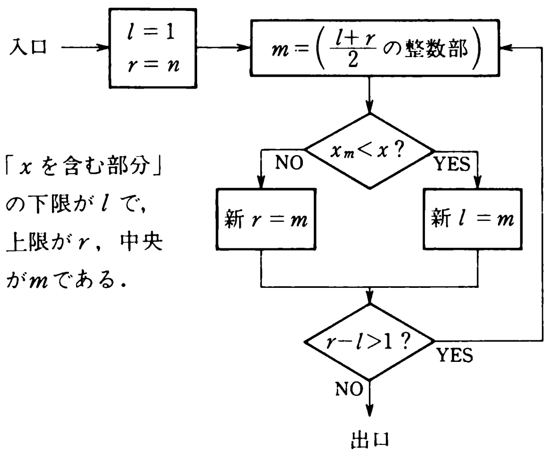
```
L=INT((X-X(0))/H
```

不等間隔の場合　最も簡単な方法は，$i=0,\ 1,\ 2,\ \cdots$の順に

$$x_i \leqq x < x_{i+1}$$

か否かを調べる（実際には右の不等式だけを調べれば十分）という方法であるが，もっと能率のよい方法として，2分法というのがある．これは，まず引数を表の中央の値（$x_{n/2}$ または $x_{(n+1)/2}$）と比較して「表の上半分にあるか下半分にあるか」を判定し，次にその上半分または下半分をまた上下に2分して，そのどちらにあるかを調べる，…という方法である．

［**流れ図**］



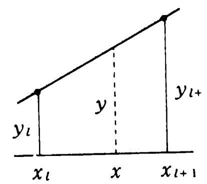
［**プログラム例**］（前後の処理は省略）

```
200 REM _ NIBUNHŌU _
210 L=1
220 R=N
230 M=INT((L+R)/2)
240 IF X(M)<Z THEN L=M ELSE R=M
250 IF R-L>1 THEN 230
```

6.12 補　　間

線形補間　2点 x_l, x_{l+1} における関数値 y_l, y_{l+1} を1次式で補間するのが線形補間である．点 x における関数値 y は次のように（近似）計算できる．

$$y = y_l + \frac{y_{l+1} - y_l}{x_{l+1} - x_l}(x - x_l)$$



また次のように表すこともできる．

$$y = \frac{y_{l+1} - y_l}{x_{l+1} - x_l}x - \frac{x_l y_{l+1} - x_{l+1} y_l}{x_{l+1} - x_l}$$

Lagrange 補間　一般に，$m+1$ 箇の点 x_0, x_1, …, x_m における関数値 y_0, y_1, …, y_m が与えられたとき，

$$P(x_i) = y_i \qquad (i = 0,\ 1,\ \cdots,\ m)$$

となるような m 次多項式を作って補間を行なうやり方を Lagrange 補間という．このような多項式は，次のようにして作ることができる．

$$P(x) = y_0 N_0(x) + y_1 N_1(x) + \cdots + y_m N_m(x)$$

ただし

$$N_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{m} \frac{(x - x_j)}{(x_i - x_j)}$$

この $P(x)$ を Lagrange 補間多項式という．

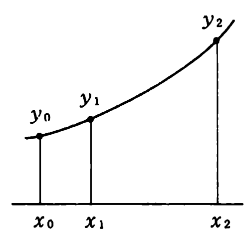
（例）　$m = 2$ の場合

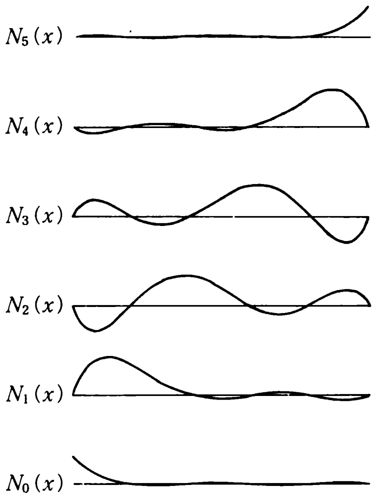
$$N_0(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)}$$

$$N_1(x) = \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)}$$

$$N_2(x) = \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)}$$

$$P(x) = y_0 N_0(x) + y_1 N_1(x) + y_2 N_2(x)$$





$N_i(x)$のグラフ
($m=5$で分点が等間隔の場合)

［**解説**］　ここに用いた関数$N_i(x)$は

分点x_iにおける値は1　　　$N_i(x_i)=1$

他の分点x_jにおける値は0　　$N_i(x_j)=0$　　$(j \neq i)$

という性質を持つ．実際，$N_i(x)$の定義式に$x=x_i$を代入すれば，すべての因子が約分できて$N_i(x)=1$になり，$x=x_j(j \neq i)$を代入すれば，分子に$(x-x_j)$という因子があるから，$N_i(x_j)=0$となる．この性質を利用して「点x_iにおける関数値だけを合わせる（他の分点には影響を与えずに）」ということができるわけである．

［**プログラム例**］（前後の処理は省略）

```
100 P=0
110 FŌR I=0 TŌ M
120     N=1
130     FŌR J=0 TŌ M
140         IF I=J THEN 160
150         N=N*(Z-X(J))/(X(I)-X(J))
160     NEXT J
170     P=Y(I)*N+P
180 NEXT I
```

6.13 数値積分

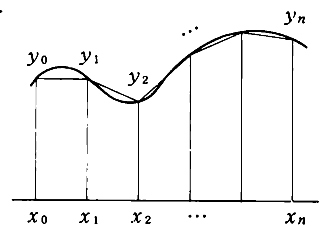
定積分 $S=\int_0^x f(x)\,dx$ の値を近似計算する方法について説明する．各種の公式があるが，利用者の立場からは，利用条件を

$f(x)$が数表として与えられている場合
$f(x)$が数式（またはプログラム）として与えられている場合

に分けて，それぞれに適した公式を覚えておくとよいと思う．

被積分関数が数表で与えられている場合

分点 x_0, x_1, x_2, …, x_n における関数値 y_0, y_1, y_2, …, y_n が与えられているとする．分点 x_0, x_n は積分の下限 a, 上限 b に一致しているとする．



台形公式　積分値 S を台形の面積の和で近似する公式である．一般形は

$$S \cong \sum_{i=1}^{n}\frac{y_{i-1}+y_i}{2}\cdot(x_i-x_{i-1})=\frac{1}{2}\sum_{i=1}^{n}(y_{i-1}+y_i)(x_i-x_{i-1})$$

特に分点が等間隔（$x_i=x_{i-1}+h$, $i=1, 2, \cdots, n$）ならば

$$S \cong \sum_{i=1}^{n}\frac{y_{i-1}+y_i}{2}\cdot h=\left(\frac{y_0}{2}+\sum_{i=1}^{n}y_i+\frac{y_n}{2}\right)\cdot h$$

となる．

［プログラム例］　$f(x)$の計算が3箇所で必要になるので，DEF文を用いるとよい．以下に示すプログラム例は，$f(x)=1/(1+x^2)$の場合であるが，DEF文をとりかえれば他の関数の計算もできる．

```
10 REM だ゛いけいこうしき
20 DEF FNC(X)=1/(1+X*X)
30 INPUT A,B,N
```

```
 40 H=(B-A)/N
 50 W=0
 60 FŌR I=1 TŌ N-1
 70     X=A+I*H
 80     W=W+FNC(X)
 90 NEXT I
100 W=FNC(A)/2+W+FNC(B)/2
110 S=W*H
120 PRINT "S=";S
```

Simpsonの公式　等間隔の3点 x_{2i}, x_{2i+1}, x_{2i+2} における関数値 y_{2i}, y_{2i+1}, y_{2i+2} をもとに，被積分関数を2次式で近似して積分を行なう．この区間における積分値は

$$\int_{x_{2i}}^{x_{2i+2}} f(x)\,dx \cong \frac{y_{2i}+4y_{2i+1}+2y_{2i+2}}{6}\cdot(x_{2i+2}-x_{2i})$$

これを $i=0$ から $i=n/2$ まで加え合わせれば S の近似値が得られる（n は偶数とする）．

［プログラム例］

```
 10 DEF FNC(X)=1/(1+X*X)
 20 INPUT A,B,N
 30 H=(B-A)/(2*N)
 40 S1=0
 50 S2=0
 60 I=1 TŌ N-1
 70     X=A+(2*I-1)*H
 80     S1=S1+FNC(X)
 90     X=A+2*I*H
100     S2=S2+FNC(X)
110 NEXT I
120 S1=S1+FNC(B-H)
130 S=(FNC(A)+4*S1+2*S2+FNC(B))*H/3
140 PRINT S
150 END
```

Newton-Cotes の公式　これは Simpson の公式を一般化したもので，等間隔の $m+1$ 点における関数値 y_{mi}，y_{mi+1}，…，y_{mi+m} をもとに，被積分関数を m 次式で近似して積分を行なう．この区間における計算式は，一般に次の形に書ける．

$$\int_{x_{mi}}^{x_{mi+m}} f(x)\,dx \cong \frac{c_0 y_{mi} + c_1 y_{mi+1} + \cdots + c_m y_{mi+m}}{C} h$$

ただし，$h = x_{mi+m} - x_{mi}$．係数は次のようになる．

m	C	c_0	c_1	c_2	c_3	c_4	c_5	c_6
2	6	1	4	1			(Simpson の公式)	
3	8	1	3	3	1			
4	90	7	32	12	32	7		
5	288	19	75	50	50	75	19	
6	840	41	216	27	272	27	216	41

[6 点公式のプログラム例]

```
 10 DATA 5,840
 20 DATA 41,216,27,272,27,216,41
 30 READ N,BUNBŌ
 40 DEF FNF(X)=1/(1+X*X)
 50 INPUT A,B
 60 H=(B-A)/N
 70 S=0
 80 FŌR I=0 TŌ N
 90     X=A+I*H
100     Y=FNF(X)
110     READ C
120     S=C*Y+S
130 NEXT I
140 S=H*S/BUNBŌ
150 PRINT "S=";S
```

被積分関数値をプログラムで計算できる場合　この場合には分点位置を自由に選べるので，

積分区間の中央付近は粗く，両端では細かくとる

粗い分割から始めて，分割をだんだん細かくしてゆく

被積分関数の性質を調べ，滑らかな部分では粗く，変動の激しい部分は細かくする

などにより能率のよい計算を行なうことができる．

第1の方法を具体化したのが**Gauss型の積分公式**で，Legendre-Gaussの公式はその代表的なものである．

Legendre-Gaussの公式　積分区間の上のLegendre多項式の零点を分点にとった公式である．このようにすると，n箇の分点を使用するだけで$2n-1$次までの多項式の厳密（離散化誤差0）な積分値を得ることができる．この公式で計算するには，下記の表から分点位置x_1，x_2，…，x_nを算定し，そこにおける被積分関数値を計算し，表の係数を掛けて加え合わせ，区間幅を掛ければよい．

$$\int_a^b f(x)\,dx = (b-a)\sum_{i=1}^{n} C_i f(x_i) \quad ただし,\ x_i = \frac{a+b}{2} + \frac{b-a}{2}t_i$$

	分点番号 i	無次元化した分点位置 t_i	係数 C_i
2点公式	1	$-\sqrt{1/3}$	1/2
	2	$+\sqrt{1/3}$	1/2
3点公式	1	$-\sqrt{3/5}$	5/18
	2	0	8/18
	3	$+\sqrt{3/5}$	5/18
4点公式	1	−0.8611363116	0.173927422
	2	−0.3399810436	0.326072577
	3	+0.3399810436	0.326072577
	4	+0.8611363116	0.173927422

6. 14 行列（マトリックス）の計算

定義 行列（マトリックス）とは数を四角に並べたもの

（例） $\begin{bmatrix} 8 & 1 & 5 \\ 2 & 9 & 3 \end{bmatrix}$ $\begin{bmatrix} 5 & 6 \\ 2 & 9 \\ 8 & 7 \\ 1 & 2 \end{bmatrix}$ 一般に $\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$ ←第1行 ←第2行 ←第m行

第1列 第2列 第n列

で，その演算は次のように定義されている．

加減算 成分ごとに加減算すること．

（例） $\begin{bmatrix} 5 & 2 & 3 \\ 4 & 0 & 7 \end{bmatrix} + \begin{bmatrix} 3 & 1 & 5 \\ 7 & 2 & 9 \end{bmatrix} = \begin{bmatrix} 5+3 & 2+1 & 3+5 \\ 4+7 & 0+2 & 7+9 \end{bmatrix} = \begin{bmatrix} 8 & 3 & 8 \\ 11 & 2 & 16 \end{bmatrix}$

スカラー倍（定数倍） すべての成分を定数倍すること．

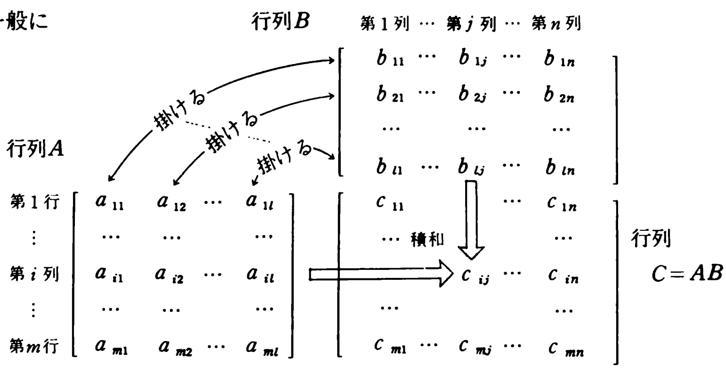
（例） $2\begin{bmatrix} 1 & 10 & 8 \\ 6 & 5 & 0 \end{bmatrix} = \begin{bmatrix} 2\times1 & 2\times10 & 2\times8 \\ 2\times6 & 2\times5 & 2\times0 \end{bmatrix} = \begin{bmatrix} 2 & 20 & 16 \\ 12 & 10 & 0 \end{bmatrix}$

積 行列 A の第 i 行と行列 B の第 j 列の積和が積 AB の第 i 行第 j 列成分となる．

（例） $\begin{bmatrix} 2 & 3 & 4 \\ 5 & 6 & 7 \end{bmatrix} \begin{bmatrix} 3 & 7 \\ 7 & 0 \\ 2 & 3 \end{bmatrix} = \begin{bmatrix} 2\times3+3\times7+4\times2 & 2\times7+3\times0+4\times3 \\ 5\times3+6\times7+7\times2 & 5\times7+6\times0+7\times3 \end{bmatrix}$

$$= \begin{bmatrix} 6+21+8 & 14+0+12 \\ 15+42+14 & 35+0+21 \end{bmatrix} = \begin{bmatrix} 35 & 26 \\ 71 & 56 \end{bmatrix}$$

一般に



計算式は，$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{il}b_{lj} = \sum_{k=1}^{l} a_{ik}b_{kj}$ $\begin{pmatrix} i = 1, \cdots, m \\ j = 1, \cdots, n \end{pmatrix}$

［**行列の乗算と入出力のプログラム例**］ 行列 A, B を入力し，その積 $C=A\cdot B$ を計算して出力するプログラムの例を示す．A は l 行 m 列，B は m 行 n 列とする．（C は当然，l 行 n 列となる）．

```
 10 DIM A(8,8),B(8,8),C(8,8)
 20 INPUT L,M,N
 30 FŌR I=1 TŌ L
 40  FŌR J=1 TŌ M
 50   PRINT "A(";I;",";J;")=";
 60   INPUT A(I,J)                        Aの入力
 70  NEXT J
 80 NEXT I
 90 FŌR I=1 TŌ M
100  FŌR J=1 TŌ N
110   PRINT "B(";I;",";J;")=";
120   INPUT B(I,J)                        Bの入力
130  NEXT J
140 NEXT I
150 FŌR I=1 TŌ L
160  FŌR J=1 TŌ N
170   S=0
180   FŌR K=1 TŌ M
190    S=A(I,K)*B(K,J)+S                  積の計算
200   NEXT K
210  C(I,J)=S
220   NEXT J
230 NEXT I
240 FŌR I=1 TŌ L
250  FŌR J=1 TŌ N                         Cの入力
260    PRINT "C(";I;",";J;")=";C(I,J)
270  NEXT J
280 NEXT I
290 END
```

6.15 連立1次方程式

Gaussの消去法　連立1次方程式の計算には普通，Gaussの消去法が用いられる．要領は筆算とだいたい同じ．それを表の形で規則的手順により計算する．

筆算のやりかた

(例) $\begin{cases} 2x+4y+6z=6 \\ 3x+8y+7z=15 \\ 5x+7y+21z=24 \end{cases}$

第1式を x について解くと

$$x=3-2y-3z$$

残りの式に代入して整理すると

$$\begin{cases} 2y-2z=6 \\ -3y+6z=9 \end{cases}$$

上の式を y について解くと

$$y=3+z$$

これを下の式に代入し整理すると

$$3z=18 \qquad \therefore \ z=6$$

順に上の式に代入して $y=9$, $x=-33$ を得る

計算機による処理法

表にまとめる

②	4	6	6
3	8	7	15
5	7	21	24

第1行を2で割り，3倍，5倍して第2行，第3行から引く．

1	2	3	3
0	②	−2	6
0	−3	6	9

第2行を2で割り，−3倍して第3行から引く．

1	2	3	3
0	1	−1	3
0	0	③	18

一般の場合　係数を表にまとめる．

$$\begin{cases} a_{11}x_1+a_{12}x_2+\cdots+a_{1n}x_n=b_1 \\ a_{21}x_1+a_{22}x_2+\cdots+a_{2n}x_n=b_2 \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ a_{n1}x_1+a_{n2}x_2+\cdots+a_{nn}x_n=b_n \end{cases}$$

a_{11}	a_{12}	$\cdots$	a_{1n}	b_1
a_{21}	a_{22}		a_{2n}	b_2
	$\cdots\cdots$			$\cdots$
a_{n1}	a_{n2}		a_{nn}	b_n

[計算手順]

前進消去

$k=1,\ 2,\ \cdots,\ n-1$ の順に

　新 $a_{kj}=a_{kj}/a_{kk}$　　$(j=k,\ k+1,\cdots,\ n)$

　新 $b_k=b_k/a_{kk}$

　$i=k+1,\cdots,\ n$ について

　　新 $a_{ij}=a_{ij}-a_{ik}a_{kj}$　　$(j=k,\ k+1,\cdots,\ n)$

　　新 $b_i=b_i-a_{ik}b_k$

$j=k+1$ からでもよい．

$$x_n=b_n/a_{nn}$$

後退代入

$k=n-1,\ n-2,\cdots,\ 1$ の順に

　$x_k=b_k-\sum_{j=k+1}^{n} a_{kj}x_j$

```
10 INPUT N
20 DIM A(N,N),X(N),B(N)
30 FŌR I=1 TŌ N                          係数と定数項の読み込み
40  FŌR J=1 TŌ N
50   PRINT "A(";I;",";J;")=";
60   INPUT A(I,J)
70  NEXT J
80  PRINT "B(";I;")=";
90  INPUT B(I)
100 NEXT I
110 FŌR K=1 TŌ N-1                         ここから前進消去
120  FŌR J=K+1 TŌ N
130   A(K,J)=A(K,J)/A(K,K)
140  NEXT J
150  B(K)=B(K)/A(K,K)
160  FŌR I=K+1 TŌ N
170   FŌR J=K+1 TŌ N
180    A(I,J)=A(I,J)-A(I,K)*A(K,J)
190   NEXT J
200   B(I)=B(I)-A(I,K)*B(K)
210  NEXT I
220 NEXT K
230 X(N)=B(N)/A(N,N)
240 FŌR K=N-1 TŌ 1 STEP -1                 ここから後退代入
250  S=0
260  FŌR J=K+1 TŌ N
270   S=S+A(K,J)*X(J)
280  NEXT J
290  X(K)=B(K)-S
300 NEXT K
310 FŌR K=1 TŌ N
320  PRINT "X(";K;")=";X(K)
330 NEXT K
340 END
```

ピボット選択　前ページのプログラムには，一つ欠点がある．それは，文130および文150において，A(K,K)の値が0のとき，計算が不可能になることである．それを避けるためには，文110と文120の間で，次のような操作をすればよい．

a_{kk}　$a_{k+1,k}$　…　a_{nk} の内で絶対値最大のものをさがす*

それが第 l 行（$l \neq k$）にあれば第 l 行と第 k 行を交換する**

このような操作を**ピボット選択という**．ピボット（pivot）とは，消去する変数 x_k の係数 a_{kk} のことである．

［**プログラム例**］　前記のプログラムの文110の後に挿入

```
111   L=K                                  第k列の最大値をさがす
112   FOR I=K+1 TO N
113     IF ABS(A(I,K))>ABS(A(L,K))
  (前行の続き) THEN L=1
114   NEXT I
115   IF L=K GOTO 125                      l = k ならばスキップ
116   FOR J=K TO N
117     SWAP A(K,J),A(L,J)
118   NEXT J                               ┐
119   SWAP B(K),B(L)                       │第k行と第
120   FOR J=K+1 TO N                       ┘l行を変換
```

---

*　脱線防止だけならば「0でないものをさがす」だけでよいが，絶対値最大のものをピボットに用いると丸め誤差を小さくする効果がある．

**　「行の変換」は「連立方程式の式の順序の変換」に相当する．あとでその影響を補正する必要はない（列の交換をした場合は補正が必要）．

掃出法（Gauss-Jordan 法）　係数と定数項を右のように表の形にまとめ，下記の手順で計算する．計算量はガウスの消去法より多いが計算手順が簡単なので小型計算機などで用いられる．

a_{11}	a_{12}	$\cdots$	a_{1n}	b_1
a_{12}	a_{22}		a_{2n}	b_2
	$\cdots\cdots\cdots\cdots$			$\cdots$
a_{n1}	a_{n2}		a_{nn}	b_n

$k=1,2,\cdots,n$ の順に
- 第 k 行を a_{kk} で割る．
- $i=1,2,\cdots,n$ について（ただし $i=k$ を除く）
  - 第 k 行の a_{ik} 倍を第 i 行から引く．

計算終了時の b_1, b_2, $\cdots$, b_n の解が x_1, x_2, $\cdots$, x_n になっている．

[**プログラム例**]　（入力の部分は129ページと同じなので省略）

```
110 FŌR K=1 TŌ N
120  P=A(K.,K)
130  FŌR J=K TŌ N
140   A(K,J)=A(K,J)/P
150  NEXT J
160  B(K)=B(K)/P
170  FŌR I=1 TŌ N
180   IF I=K GŌTŌ 240
190   Q=A(I,K)
200   FŌR J=K TŌ N
210    A(I,J)=A(I,J)-Q*A(K,J)
220   NEXT J
230   B(I)=B(I)-Q*B(K)
240  NEXT I
250 NEXT K
260 FŌR I=1 TŌ N
270  PRINT "X(";I;")=";B(I)
280 NEXT I
290 END
```

6.16 逆 行 列

定義 正方行列を A, 単位行列を I とするとき, $AX=I$ になる行列 X を A の逆行列といい A^{-1} で表す.

計算法（原理） 逆行列を求めるには要するに

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & & x_{2n} \\ \cdots & \cdots & & \cdots \\ x_{n1} & x_{n2} & \cdots & x_{nn} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \cdots & \cdots & & \cdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$

になるような $x_{ij}(i,j=1,2,\cdots,n)$ を求めればよく，それには連立１次方程式 $A\boldsymbol{x}_j=\boldsymbol{e}_j$ （$\boldsymbol{x}_j$は X の第 j 列, $\boldsymbol{e}_j$ は I の第 j 列）を $j=1,2,\cdots,n$ について解けばよい.

計算法（実際） 連立１次方程式の解法としては掃出法が簡単でよい（連立方程式を，１題解くだけならば Gauss の消去法の方が速いが，逆行列の計算の際には掃出法でも Gauss の消去法でも計算時間は同じになる）. 手順はまず

a_{11}	a_{12}	$\cdots$	a_{1n}	1	0	$\cdots$	0	第１行
a_{21}	a_{22}		a_{2n}	0	1		0	第２行
$\cdots$	$\cdots$		$\cdots$	$\cdots$	$\cdots$		$\cdots$	
a_{n1}	a_{n2}	$\cdots$	a_{nn}	0	0	$\cdots$	1	第 n 行

という表を作り,

- $k=1,\ 2,\ \cdots,\ n$ の順に
  - 第 k 行を a_{kk} で割る
  - $i=1,\ 2,\ \cdots,\ n$ に対し（ただし $i=k$ を除く）
    - 第 i 行から第 k 行の a_{ik} 倍を引く.

という計算をする. その終了時点に，表の左半分は I になり，右半分は A^{-1} になっている.

記憶場所の節約 上記の計算の中間結果は，第 k 段階の開始時点について考えてみると，配列 A の第 k 列より左側は単位行列と同じ形になっていて，実際にはもはや計算の必要はなく，また表の右半分の第 k

列より右側は，実質的な計算はまだ行なわれておらず，単位行列の形がそのまま残っている．そこで，このように内容が自明な部分を除き，配列Aの第 k 列より左には A^{-1} を入れるようにすれば記憶場所が節約され，計算時間も半分で済む．次のプログラムは，そのような方式をとっている．

［プログラム例］

```
 10 INPUT N
 20 DIM A(N,N)                              データー入力
 30 FŌR I=1 TŌ N
 40  FŌR J=1 TŌ N
 50   PRINT "A(";I;",";J;")";
 60   INPUT A(I,J)
 70  NEXT J
 80 NEXT I
 90 FŌR K=1 TŌ N                            ここから逆行列の計算
100  P=A(K,K)
110  A(K,K)=1
120  FŌR J=1 TŌ N
130   A(K,J)=A(K,J)/P
140  NEXT J
150  FŌR I=1 TŌ N
160  IF I=K GŌTŌ 220
170   Q=A(I,K)
180   A(I,K)=0
190   FŌR J=1 TŌ N
200    A(I,J)=A(I,J)-Q*A(K,J)
210   NEXT J
220  NEXT I
230 NEXT K
240 FŌR I=1 TŌ N                            結果の出力
250  FŌR J=1 TŌ N
260   PRINT "A(";I;",";J;")=",A(I,J)
270  NEXT J
280 NEXT I
```

6.17 固有値，固有ベクトル

与えられた正方行列 A に対し

$$A\boldsymbol{x} = \lambda\boldsymbol{x} \qquad \text{ただし } \boldsymbol{x} \neq 0$$

を満たすベクトル $\boldsymbol{x}$（**固有ベクトル**という）とスカラー $\boldsymbol{\lambda}$（**固有値**という）を求める問題を，行列の固有値問題という．このような計算は，固有振動数の計算や主成分分析（統計解析の一手法）などの際に必要になる．A が実対称行列の場合とそうでない場合とでは，問題の性質も解法もかなり異なる．応用では実対称の場合が多く，実対称ならば計算が簡単なので，以下では実対称行列の固有値問題の解法について説明する．

Jacobi 法　実対称行列の固有値問題の解法は数多くあるが，最も簡単で，そのわりに能率がよく，安全確実なのは Jacobi 法である．

［**原理**］ 実対称行列 A の非対角要素の一つを a_{pq} とし，

$$M = \begin{bmatrix} \ddots & & & & \\ & c & & s & \\ & & \ddots & & \\ & -s & & c & \\ & & & & \ddots \end{bmatrix} \begin{matrix} \\ \leftarrow p\text{ 列目} \\ \\ \leftarrow q\text{ 列目} \\ \\ \end{matrix}$$

（対角上に 1 を並べる，他の部分は 0；c の列は ↑ p 行目，↑ q 行目）

ただし，

$$\theta = \frac{1}{2}\arctan\frac{2a_{pq}}{a_{pp}-a_{qq}}$$

$$c = \cos\theta$$

$$s = \tan\theta$$

という行列 M を作って，新 $A = MAM^{\mathrm{T}}$ という変換をすると，

新 $a_{pq} = 0$　　新 $a_{qp} = 0$　　になる

新 A はやはり実対称で，A と同じ固有値をもつ

という性質がある．このような変換をくりかえして，非対角要素をすべて 0（または非常に小さな値）にすれば，そのとき固有値は新 A の対角上に並ぶ．また固有ベクトルは，変換に用いた M^{T} の積

$$V = M_m^{\mathrm{T}} M_{m-1}^{\mathrm{T}} \cdots M_2^{\mathrm{T}} M_1^{\mathrm{T}}$$

の各列として得られる．p，q の選び方はいろいろあるが，右のプログラムでは「その段階の非対角要素の中で絶対値最大のものを a_{pq} とする」という方式（古典的方法）を用いている．この方法は，最大値をさがすのに手間がかかるが，必要な反復回数は最も少ない．

［**計算手順**］ $V=I$（単位行列）として出発し，すべての非対角要素の絶対値が十分小さくなるまで，下記の手続をくりかえす．

1） p，q を決める

2） $\theta=\dfrac{1}{2}\arctan\dfrac{2a_{pq}}{a_{pp}-a_{qq}}$ （ただし $a_{pp}=a_{qq}$ の場合は $\theta=\pi/4$ とする）

3） $c=\cos\theta$　　$s=\tan\theta$

4） 新 $A=MAM^{\mathrm{T}}$ の乗算を行なう．実際には第 p 行，第 q 行，第 p 列，第 q 列の計算だけをすれば十分である（他の部分は不要）．

5） 新 $V=VM^{\mathrm{T}}$ の乗算を行なう．実際には第 p 列と第 q 列の計算だけをすれば十分である（他の部分は不要）．

［**参考用**］ プログラム・テストのための行列

$$(\mathrm{i})\quad \begin{bmatrix} 5 & 4 & 1 & 1 \\ 4 & 5 & 1 & 1 \\ 1 & 1 & 4 & 2 \\ 1 & 1 & 2 & 4 \end{bmatrix} \qquad (\mathrm{ii})\quad \begin{bmatrix} 1111 & -1089 & 891 & 909 \\ 1089 & 1111 & -909 & 891 \\ 891 & -909 & 1111 & -1089 \\ 909 & 891 & -1089 & 1111 \end{bmatrix}$$

厳密な固有値，固有ベクトルは次のようになる．

（i）　$\lambda_1=10$　　$\lambda_2=5$　　$\lambda_3=2$　　$\lambda_4=1$

$$\boldsymbol{v}_1=\begin{bmatrix}2\\2\\1\\1\end{bmatrix}\quad \boldsymbol{v}_2=\begin{bmatrix}-1\\-1\\2\\2\end{bmatrix}\quad \boldsymbol{v}_3=\begin{bmatrix}0\\0\\-1\\1\end{bmatrix}\quad \boldsymbol{v}_4=\begin{bmatrix}-1\\1\\0\\0\end{bmatrix}$$

（ii）　$\lambda_1=4$　　$\lambda_2=40$　　$\lambda_3=400$　　$\lambda_4=4000$

$$\boldsymbol{v}_1=\begin{bmatrix}1\\1\\1\\1\end{bmatrix}\quad \boldsymbol{v}_2=\begin{bmatrix}1\\1\\-1\\-1\end{bmatrix}\quad \boldsymbol{v}_3=\begin{bmatrix}1\\-1\\-1\\1\end{bmatrix}\quad \boldsymbol{v}_4=\begin{bmatrix}1\\-1\\1\\-1\end{bmatrix}$$

［**注意**］ 固有ベクトルの計算結果は，普通，上記と同じにはならないが，成分の比が上記に一致すればよい．

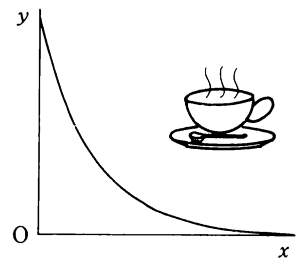
```
7000 REM ########################
7010 REM # ヤコヒ゛ほうによる こゆうちけいさん #
7020 REM #  serial Jacobi ほう  #
7030 REM #  みき゛うえはんふ゛んて゛けいさん  #
7040 REM ########################
7050 REM しゅうんくはんていし゛ょうすうをせってい
7060 CMAX=0
7070 FOR I=1 TO N
7080 FOR J=I TO N
7090 IF ABS(A(I,J))<CMAX GOTO 7120
7100 P=I : Q=J
7110 CMAX=ABS(A(I,J))
7120 NEXT J
7130 NEXT I
7140 EPS=CMAX*0.0001
7150 KMAX=100
7160 REM こゆうへ゛クトルのけいさんのし゛んひ゛
7170 FOR I=1 TO N
7180 FOR J=1 TO N
7190 V(I,J)=0
7200 NEXT J
7210 V(I,I)=1
7220 NEXT I
7230 REM ・・・・・・・ [メイン・ループ゜] ・・・・・・・
7240 FOR K=1 TO KMAX
7250 REM たいかくに ちかいところから へんかんを おこなう
7260 FOR L=1 TO N-1
7270 REM き゛ょうは゛んこ゛うし゛ゅんにスキャン
7280 PMAX=N-L
7290 FOR P=1 TO PMAX
7300 Q=P+L
7310 REM ヒ゜ホ゛ットは P,Q
7320 IF ABS(A(P,Q))<EPS GOTO 7730
7330 PRINT K;P;Q;
7340 PRINT TAB(15);A(P,Q);
7350 PRINT TAB(31);TIME$
7360 APP=A(P,P)
7370 AQQ=A(Q,Q)
7380 APQ=A(P,Q)
7390 ALPHA=(APP-AQQ)*0.5
7400 BETA=SQR(ALPHA*ALPHA+APQ*APQ)
7410 C=SQR((1+ABS(ALPHA)/BETA)*0.5)
7420 S=0.5*APQ/(BETA*C)
7430 IF ALPHA<0 THEN S=-S
7440 CC=C*C : SS=S*S : CS=C*S
```

```
7450 A(P,P)=APP*CC+2*APQ*CS+AQQ*SS
7460 A(Q,Q)=APP*SS-2*APQ*CS+AQQ*CC
7470 A(P,Q)=0
7480 IF P=1 GOTO 7540
7490 FOR I=1 TO P-1
7500 AIP=A(I,P) : AIQ=A(I,Q)
7510 A(I,P)=C*AIP+S*AIQ
7520 A(I,Q)=C*AIQ-S*AIP
7530 NEXT I
7540 IF Q=N GOTO 7600
7550 FOR J=Q+1 TO N
7560 APJ=A(P,J) : AQJ=A(Q,J)
7570 A(P,J)=C*APJ+S*AQJ
7580 A(Q,J)=C*AQJ-S*APJ
7590 NEXT J
7600 IF L=1 GOTO 7660
7610 FOR I=P+1 TO Q-1
7620 API=A(P,I) : AIQ=A(I,Q)
7630 A(P,I)=C*API+S*AIQ
7640 A(I,Q)=C*AIQ-S*API
7650 NEXT I
7660 REM こゆうべクトルのけいさん
7670 FOR I=1 TO N
7680 VIP=V(I,P)
7690 VIQ=V(I,Q)
7700 V(I,P)=C*VIP+S*VIQ
7710 V(I,Q)=C*VIQ-S*VIP
7720 NEXT I
7730 NEXT P
7740 NEXT L
7750 REM しゅうそく はんてい
7760 FOR I=1 TO N-1
7770 FOR J=I+1 TO N
7780 IF ABS(A(I,J))>EPS GOTO 7820
7790 NEXT J
7800 NEXT I
7810 RETURN : REM --- normal exit ---
7820 NEXT K
7830 REM ･･･[メイン･ループ]･･･(ここまで)･･･
7840 PRINT "はんぷく うちきり"
7850 BEEP : STOP
7860 RETURN
7870 REM --- end of JACOBI ---
```

6.18 常微分方程式

コンピューターによる常微分方程式の計算法は，単独(非連立)，1階の常微分方程式の初期値問題の解法が基本的で，それを応用すれば，連立の場合や高階の場合の計算も行なうことができる．

簡単な例題　はじめに，簡単な例題によって，基本的な考え方を説明しておくことにする．



コップに100℃の湯を入れ，室温0℃の部屋に置く．湯の温度を y ℃，時刻を x とする．湯の温度の変化率は（ごく単純化して考えれば）湯の温度と室温の差に比例するであろう．その比例定数は，コップの熱伝導率，表面積，比熱，その他いろいろなものに関係するが，ここでは話を簡単にするため，比例定数は1として計算することにする（一般の場合でも，いわゆる無次元化という処理をすれば同じ形になる）．以上の関係を式で書けば

$$\text{微分方程式}\quad \frac{dy}{dx} = -y \qquad \text{初期条件}\quad y(0) = 100$$

となる．このような関係を満たす関数 $y(x)$ を求める問題が，常微分方程式の初期値問題（の簡単な一例）である．

簡単な解法　上記の式の dy/dx を間隔 h の差分商で代用すれば

$$\frac{y(x+h)-y(x)}{h} = -y(x)$$

移項すれば

$$y(x+h) = y(x) - h\cdot y(x)$$

となる．これを「現在時刻 x における温度が既知のとき，その h 時間後の温度 $y(x+h)$ を近似計算する式」として用いれば，初期値 $y(0)$ から出発して，$y(x+ih)$ $(i=1, 2, 3, \cdots)$ を求めることができる．

［**プログラム例**］ 時刻 $x=1$ までの計算を行なうことにする．

```
10 INPUT H
20 N=1/H                        総ステップ数を計算
30 X0=0                         } 初期値の設定
40 Y=100                        }
50 FŌR I=0 TŌ N
60      X=X0+I*H                } 現時刻と現時点にお
70      PRINT X,Y               } ける温度を出力
80      Y=Y-H*Y                 次の時点の温度を計算
90 NEXT I
```

［**注意**］ 上記のプログラムで実習する際には，ひとまず $h=0.1$ で計算してプログラムが正常に動くことを確認し，次に $h=0.01$ の場合を計算して，前の計算結果と比較するとよい．h をもっと小さくする場合には，上記のプログラムだと出力が多くなりすぎるので，次のようにして出力を間引く必要がある．

```
45 M=INT(0.1/H+0.5)
50 FŌR I=0 TŌ 10                } 
60      X=X0+I*0.1              } 0.1間隔で出力
70      PRINT X,Y               }
75      FŌR J=1 TŌ M            }
80          Y=Y-H*Y             } 出力せずに進める
85      NEXT J                  }
90 NEXT I
```

なお，h は0.1の整数分の1にとるものとする．

［**補足**］ 左記の計算式は，$y(x+h)=(1-h)y(x)$ と書くこともできる．これより，

$$y(1)=y(x+nh)=(1-h)^n y(0)=100(1-1/n)^n$$

これの $n\to\infty$ の極限値は，$100e^{-1}$ となり，厳密解の値に一致する．

6. 19 Euler 法

単独，1階の常微分方程式は，一般に

$$y' = f(x, y) \tag{1}$$

と書ける．ただし，x は独立変数，$y(x)$ は未知関数，y' は dy/dx の意味，$f(x, y)$ は「x と y の値が与えられたときに y' の値を求めるための計算式(必ずしも一つの式で表せなくてもよく，要するに x と y から y' の値が一意的に計算できればよい)」である．コンピューターで数値的に計算できるのは，一つの（与えられた）初期条件

$x = x_0$ のとき，$y = y_0$

から出発して，

$$x_0,\ x_1,\ x_2,\ \cdots,\ x_i,\ \cdots$$

（ただし，$x_i = x_{i-1} + h$，hは分点間隔）における解 y の近似値

$$y_0,\ y_1,\ y_2,\ \cdots,\ y_i,\ \cdots$$

を求めることである．このような計算をするには，基本的には，「1ステップ進めるための技術」すなわち「$x = x_i$ のときに $y = y_i$ である，ということがわかったときに，$x = x_{i+1}$ における y の値 y_{i+1} を計算する方法」を確立すればよい．

[計算の方法]　前節と同様に，微分方程式(1)の微分商を差分商に置き換えれば

$$\frac{y(x+h) - y(x)}{h} = f(x, y)$$

これより

$$y(x+h) = y(x) + h \cdot f(x, y) \tag{2}$$

すなわち，

$$y_{i+1} = y_i + h \cdot f(x_i, y_i)$$

という計算法が導かれる．これを **Euler 法**という．

（例）　微分方程式 $y' + y = e^x$ の場合，$f(x, y) = e^x - y$ であるから，計算式は，$y_{i+1} = y_i + h(e^x - y)$ となる．

[**解説**] 「微分を差分で代用する」というと乱暴なようであるが，関数 y を点 x のまわりの Taylor 展開

$$y(x+h)=y(x)+\frac{h}{1!}y'(x)+\frac{h^2}{2!}y''(x)+\cdots+\frac{h^n}{n!}y^{(n)}(x)+\cdots$$

の 2 次以上の項を無視すれば式(2)が得られるし，式(1)を x から $x+h$ まで積分して得られる関係式

$$y(x+h)=y(x)+\int_x^{x+h}f(x,\ y)\,dx$$

の被積分関数 f を，定数 (xにおける値) で近似しても式(2)が得られるわけで，さらに「分割を限りなく細かくするとき，Euler 法の結果は厳密解に収束する」ということも証明されている．

[**プログラム例**]

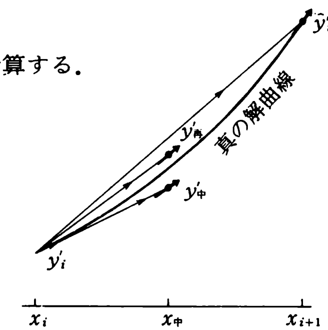
```
 10 INPUT X0,Y,H,N
 20 FŌR I=0 IŌ N
 30     X=X0+I*H
 40     PRINT X,Y
 50     GŌSUB 100
 60     Y=Y+H*F
 70 NEXT I
 80 STŌP
100 REM *** F(X,Y) NŌ KEISAN ***
110 F=X*Y                        f(x, y)の計算式の一例
120 RETURN
```

/

6.20 Runge-Kutta 法

Euler 法は簡単で便利であるが，精度はあまりよくない．そこで，精度を改良した解法が数多く開発された．その中で，比較的簡単でありながら精度がよく安全確実な公式として最もよく使われているのが，以下で説明する Runge-Kutta 法である．

［**計算の手順**］ 微分方程式は $y'=f(x,\ y)$ とし，その $x=x_i$ における y の値 y_i までは計算済みとする．次の値（$x_{i+1}=x_i+h$ における y の値 y_{i+1}）は，下記のようにして計算する．



1) x_i と x_{i+1} の中点 $x_{中}=x_i+h/2$ における y の値 $y_{中}$ を Euler 法により計算する．

$$y_{中}=y_i+(h/2)y'_i$$

2) それをもとに，中点における勾配 $y'_{中}$ を計算する．

$$y'_{中}=f(x_{中},\ y_{中})$$

3) 上記の $y'_{中}$ を用いて，中点の値を再計算する．

$$y_{再}=y_i+(h/2)\ y'_{中}$$

4) それをもとに，中点における勾配を再計算する．

$$y'_{再}=f(x_{中},\ y_{再})$$

5) それを用いて，x_{i+1} における y の値の概算値 $\hat{y}_{i+1}$ を作る．

$$\hat{y}_{i+1}=y_i+hy'_{再}$$

6) それをもとに，y'_{i+1} の概算値 $\hat{y}'_{i+1}$ を求める．

$$\hat{y}'_{i+1}=f(x_{i+1},\ \hat{y}_{i+1})$$

7) 以上，4 回計算された勾配 y'_i，$y'_{中}$，$y'_{再}$，$\hat{y}'_{i+1}$ を，1：2：2：1 の重みで平均して用いて最終的な y_{i+1} の値を計算する．

$$y_{i+1}=y_i+\frac{y'_i+y'_{中}+y'_{再}+\hat{y}'_{i+1}}{6}\,h$$

こうして計算される y_{i+1} と真の $y(x_{i+1})$ の差は h^5 のオーダーである（要するに非常に小さい）ことが証明されている．

［**数値例**］ 6.18節と同じ問題

$$y' = -y \qquad y(0) = 100$$

を Runge-Kutta 法で $h = 1$ として計算してみる．

$$x_{中} = 0.5 \qquad y_{中} = 100 + (-100)/2 = 50 \qquad y'_{中} = -50$$

$$y_{再} = 100 + (-50)/2 = 75 \qquad y'_{中} = -75$$

$$\hat{y}_{i+1} = 100 - 75 = 25 \qquad \hat{y}'_{i+1} = -25$$

$$y_{i+1} = 100 + \frac{-100 - 50 \times 2 - 75 \times 2 - 25}{6} \times 1 = 37.5$$

その値は，$100e^{-x}$の Taylor 展開式の第5項までの和の $x = 1$ の値

$$100\left(1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + \frac{1}{4!}\right) = 37.5$$

に等しい．

［**プログラム例**］

```
 10 DEF FNC(Y)=        ……………………… これは利用者が定義
 20 INPUT X0,Y,H,N
 30 FŌR I=0 TŌ N
 40     PRINT X0+I*H,Y
 50     F13FNC(X,Y)                  y'_i
 60     X=X+H/2
 70 F2=FNC(X,Y+H*F1/2)               y'中
 80 F3=FNC(X,Y+H*F2/2)               y'再
 90      X=X+H/2
100 F4=FNC(X,Y+H*F3)                 ŷ'_{i+1}
220 F=(F1+2*F+2*F3+F4)/6
120 Y=Y+F*H                          y_{i+1}
130 NEXT I
```

6.21 高階および連立微分方程式の解法

n 階微分方程式の場合，2 階から $n-1$ 階までの導関数をそれぞれ別個の関数として扱えば，連立 1 階の微分方程式になる．

（例 1） $y''+2y'+3y = \cos x$ の場合，y' を z と置けば，

$$\begin{cases} y' = z \\ z' = y'' = \cos x - 2z - 3y \end{cases}$$

（例 2） $y'''+yy'' = 0$ の場合，$y_1 = y$，$y_2 = y'$，$y_3 = y''$ と置けば

$$\begin{cases} y'_1 = y_2 \\ y'_2 = y_3 \\ y'_3 = -y_1y_3 \end{cases}$$

このように連立 1 階の形にしてから，次に述べる方法を適用する．

連立 1 階微分方程式の解法　連立 1 階の常微分方程式は，一般に次の形に書ける．

$$\begin{cases} y'_1 = f_1(x, y_1, y_2, \cdots, y_n) \\ y'_2 = f_2(x, y_1, y_2, \cdots, y_n) \\ \qquad\cdots\cdots \\ y'_n = f_n(x, y_1, y_2, \cdots, y_n) \end{cases}$$

ただし，今回の添字は「変数の番号」を表している．

連立の場合の計算法は，本質的には単独の場合の計算法と同じで，ただそれを n 箇の変数について同時に計算すればよい．

Euler 法を適用する場合の計算式は次のようになる．

$$\begin{cases} y_1(x+h) = y_1(x)+hf_1(x, y_1(x), \cdots, y_n(x)) \\ \qquad\cdots\cdots \\ y_n(x+h) = y_n(x)+hf_n(x, y_1(x), \cdots, y_n(x)) \end{cases}$$

（例） 左記の例1の場合なら，次のように計算すればよい．

$$\begin{cases} y(x+h) = y(x) + hz \\ z(x+h) = z(x) + h(\cos x - 2z - 3y) \end{cases}$$

初期条件を $y(0) = 0$, $z(0) = 0$ とし，進み幅を $h = 0.1$ とすれば

$$\begin{cases} y(0.1) = 0 + 0.1 \times 0 = 0 \\ z(0.1) = 0 + 0.1 \times (1 - 2 \times 0 - 3 \times 0) = 0.1 \end{cases}$$

$$\begin{cases} y(0.2) = 0 + 0.1 \times 0.1 = 0.01 \\ z(0.2) \fallingdotseq 0.1 + 0.1 \times (0.995 - 2 \times 0.1 - 3 \times 0) \fallingdotseq 0.1795 \end{cases}$$

以下同様．コンピューターで計算するのであれば，次のようにする．

```
10 INPUT X0,Y,Z,K,H
20 FOR L=0 TO K
30     X=L0+L*H
40     PRINT X,Y,Z
50     Y0=Y+H*Z
60     Z=Z+H*(COS(X)-2*Z-3*Y)
70     Y=Y0
80 NEXT L
```

（ただしKは終了までのステップ数）

Runge–Kutta 法を適用する場合の計算式は次のようになる．

$$y'_{i左} = f_i\,(x_{左},\ y_{1左},\ y_{2左},\ \cdots,\ y_{n左}) \qquad (i = 1, 2, \cdots, n)$$

$$y_{i中} = y_{i左} + (h/2)\, y'_{i左} \qquad (\ 〃\ )$$

$$y'_{i中} = f_i\,(x_{中},\ y_{1中},\ y_{2中},\ \cdots,\ y_{n中}) \qquad (\ 〃\ )$$

$$y_{i再} = y_{i左} + (h/2)\, y'_{i中} \qquad (\ 〃\ )$$

$$y'_{i再} = f_i\,(x_{中},\ y_{1再},\ y_{2再},\ \cdots,\ y_{n再}) \qquad (\ 〃\ )$$

$$y_{i右} = y_{i左} + h y'_{i再} \qquad (\ 〃\ )$$

$$y'_{i右} = f_i\,(x_{右},\ y_{1右},\ y_{2右},\ \cdots,\ y_{n右}) \qquad (\ 〃\ )$$

$$y_{i次} = y_{i左} + \frac{y'_{i左} + 2y'_{i中} + 2y'_{i再} + y'_{i右}}{6}\, h \qquad (\ 〃\ )$$

ただし，$x_{中} = x_{左} + h/2$, $x_{右} = x_{左} + h$, また $x_{右}$ と $y_{i次}$ が次のステップの $x_{左}$ と $y_{i左}$ になる．

[プログラム例]

```
 10 DIM Y(5),Y0(5),F(5)
 20 DIM F1(5),F2(5),F3(5)
 30 READ X0,N,H,K
 40 FŌR I=1 TŌ N
 50     READ Y0(I)
 60     Y(I)=Y0(I)
 70 NEXT I
 80 FŌR L=0 TŌ K
 90     X=X0+L*H
100     GŌSUB 500
110     FŌR I=1 TŌ N
120         F1(I)=F(I)
130         Y(I)=YŌ(I)+F(I)*H/2
140     NEXT I
150     X=X+H/2
160     GŌSUB 500
170     FŌR I=1 TŌ N
180         F2(I)=F(I)
190         Y(I)=Y0(I)+F(I)*H/2
200     NEXT I
210     GŌSUB 500
220     FŌR I=1 TŌ N
230         F3(I)=F(I)
240         Y(I)=Y0(I)+F(I)*H
250     NEXT I
260     X=X+H/2
270     GŌSUB 500
280     FŌR I=1 TŌ N
290         Y(I)=Y0(I)+(F1(I)+
   (続き)     └─2*F2(I)+2*F3(I)+F(I))*H/6
300         Y0(I)=Y(I)
310         PRINT Y(I),
320     NEXT I
```

```
330     PRINT X              xの値は一番右の列に印刷される
340 NEXT L
350 END
500 REM                      fの計算のプログラムを利用者が書く
510 F(1)=Y(2)                (これは例1の計算の場合)
520 F(2)=COS(X)-2*Y(2)-3*Y(1)
530 RETURN
540 DATA 0,2,0.1,100
550 DATA 0                   初期条件 { y(0) = 0
560 DATA 0                            { y'(0) = 0
570 END
```

常微分方程式の数値計算の注意　Runge–Kutta 法は，1ステップ進む間（x から $x+h$ までの間）各未知関数の4階までの導関数が連続であることを仮定している．したがって，その間に f の計算の不連続点を作らないよう（たとえば，自動制御系のシュミレーションの場合，スイッチのON，OFFをしないように）注意する必要がある．

進み幅 h は小さすぎても大きすぎてもいけない．小さすぎると

$$y(x+h) = y(x) + (\text{増分})$$

の計算のとき，右図のようになって，増分が無視されてしまう．一方，進み幅が大きすぎると精度が悪くなり，さらに過大にすれば発散することがある．

	×.×××××	$y(x)$
+	0.000000××	増分
	×.×××××	$y(x+h)$
	←1ワード→	

6.22 擬似乱数による数値実験

関数 RND　統計的現象の「実験」をコンピューターで行うことができる．そのための道具としてRNDという関数が用意されている．

書き方は機種によって多少異なり，単にRNDと書けばよい機種と，RND(1)のように形式的な引数を付ける機種とがある．以下のプログラム例は前者の方式で書いてある．

関数RNDは，$0 \leqq \mu < 1$ の範囲の一様乱数を作り出す．すなわち，関数RNDを呼ぶたびに，上記の範囲の値 μ がランダム（random，無作為，不規則，でたらめ）に出て来る（一様分布に従って）．

実際にどんな数が出て来るのか，次のようなプログラムを作って，実行させてみよう．

［乱数列を出力するプログラム］

```
10 FOR I=1 TO 20
20     U=RND
30     PRINT U
40 NEXT I
50 END
```

乱数発生のしくみ　乱数を作るには，サイコロを振るように，物理現象を用いるのが理想的であるが，コンピューターの中でそう簡単に良質の物理乱数を作ることはできないので，普通は，たとえば

N_{i+1} = (N_i を K 倍して M で割った余り)

$$u_{i+1} = N_{i+1}/M$$

（ただし，N, M, K はいずれも正の整数）というような方法により，「乱数に似た性質をもつ数列」を作り出して使用する．これを擬似乱数という．これは規則的に作り出すのであるから，完全にランダムとはいえないが，うまく作れば（たとえば上記の方法の場合，定数 K、M と初期値 N_0 を適切に選べば）初等的数値実験の目的には十分使える，ということが知られている．

RANDOMIZE 文　擬似乱数は，数列としては乱数に近い性質をもつが，同じ初期値 N_0 を使えば同じ数列が出てしまう．それを避けるにはプログラムの最初にRANDŌMIZEと書けばよい．

サイコロに変換　関数RNDで得られるのは $0 \leqq u < 1$ の範囲の一様乱数だけであるが，これを適当に変換することにより，上記以外の分布に従う乱数を作ることができる．

たとえば，これを「1から6の目が等確率で出るサイコロ」の目の実数値に換算するには，

$0 \leqq u < 1/6$　ならば　⚀

$1/6 \leqq u < 2/6$　ならば　⚁

$5/6 \leqq u < 1$　ならば　⚅

とすればよいであろう．BASIC でこのような変換を行なうには，次のように書くと簡単でよい．

```
S=INT(6*U)+1
```

乱数の発生と変換を兼ねて次のように書くことができる．

```
S=INT(6*RND)+1
```

［**使用例**］ サイコロを2回ずつ100組（計200回）振って，2回の目数の和がちょうど7になる組数を数えてみる．

```
10 N=0                                  カウンターのリセット
20 FŌR K=1 TŌ 100
30      S1=INT(6*RND)+1     ┐サイコロを
40      S2=INT(6*RND)+1     ┘2回振る
50      IF S1+S2<>7 GŌTŌ 80             和≠7?
60      N=N+1                           カウント・アップ
70 NEXT K
80 PRINT N;
```

関数 RND の応用 ——（算数練習機）　0から9までの乱数を二つ作り（I，Jとする），

Iの値　プラス印　Jの値　イコール印

を表示し，生徒に答を入力させ（それをKとする），

I+J=K　ならば　Ō印

I+J≠K　ならば　X印

を出力するプログラムを作ってみよう．0から9までの乱数を作るには

INT(10*RND)

とすればよい．

[プログラム例]

```
10 I=INT(10*RND)
20 J=INT(10*RND)
30 PRINT I;"+";J;"=";
40 INPUT K
50 IF I+J=K THEN PRINT"Ō" ELSE
60 PRINT                       (前行の続き) PRINT"X"
70 GŌTŌ 10
```

[出力例]

```
 5 + 2 = 7
O

 5 + 6 = 11
O

 1 + 8 = 7
X

 0 + 9 = 9
O

 3 + 5 = 8
O

 4 + 0 = 4
O

 5 + 5 = 11
X

 3 + 4 = 7
O
```

6.23 正 規 乱 数

平均値 0，分散 1 の正規分布 $N(0,1)$ に従う乱数 y を作るには，「$0 \leqq u < 1$ の一様乱数 u を12箇加えて 6 を引く」という方式

$$y = u_1 + u_2 + \cdots + u_{12} - 6$$

が最も簡単である．y が（近似的に）$N(0,1)$ に従うことは，中心極限定理による．

［プログラム例］ 上記の方法で正規乱数を100個作り，そのヒストグラムを描くプログラムを作ってみよう．ヒストグラムの範囲と間隔は，$-3 \leqq y < +3$, $\Delta y = 0.5$ とする．

```
 10 DIM H(12)
 20 FŌR I=1 TŌ 12  ┐
 40  H(I)=0        ├ カウンターをクリア
 50 NEXT I         ┘
 60 FŌR K=1 TŌ 100                  100回反復
 70  Y=-6
 80  FŌR J=1 TŌ 12  ┐
 90   Y=Y+RND       ├ 正規乱数を作る
 90  NEXT J         ┘
100  I=INT(2*Y+6)
110  IF I<1 GŌTŌ 150
120  IF I>12 GŌTŌ 150
130  H(I)=H(I)+1
140 NEXT K
150 FŌR I=1 TŌ 12
160  IF H(I)=0 GŌTŌ 210
170   FŌR L=I TŌ H(I)
180    PRINT "*";
190   NEXT L
200  PRINT
210 NEXT I
```

出力例

```
*
****
*******
****************
**********************
*****************
******************
****
*****
****
*
```

6.24 指 数 乱 数

確率密度関数 $\lambda e^{-\lambda x}$ の指数分布（その平均値は λ^{-1}，分散は λ^{-2} である）に従う乱数を作るには，区間［0, 1］上の一様乱数 u を

$$x = -\lambda^{-1}\log_e u$$

で変換すればよい．BASIC では，これを利用者定義関数の形

```
DEF FNE=-M*LOG(RND)
```

にして用いると便利である（ただし，$M = \lambda^{-1} =$（平均値））．

［用途］ 単位時間内に平均 λ 人がランダムに窓口に到着するとき，その到着間隔（1人が来てから次の人が来るまでの時間）は上記の分布に従う．そのため，これは待ち行列のシミュレーションによく用いられる．

Poisson 乱数　単位時間内に平均 λ 人がランダムに窓口に到着するとき，単位時間内に実際に到着する人数 n の分布は，平均値 λ の Poisson 分布

$$P(n) = e^{-\lambda}\frac{\lambda^n}{n!}$$

に従う．このような乱数 n を作るには，普通，次のようにする．

区間［0, 1］上の一様乱数 $u_1, u_2, u_3, \cdots$ を作る．

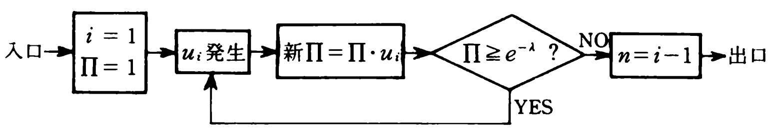
それを順に掛けてゆく．$u_1, u_1u_2, u_1u_2u_3, \cdots$

積が $e^{-\lambda}$ より小さくなったら止める．

その直前までの箇数を n とする．すなわち

$$u_1u_2\cdots u_n \geqq e^{-\lambda} > u_1u_2\cdots u_{n+1}$$

［解説］ これは，指数乱数（いいかえれば「到着間隔」）$x_1, x_2, \cdots$ を発生させて，その和が 1（すなわち単位時間）になるまでの人数 n を数えるのと本質的に同じことで，その「log の和」をとるかわりに「積の log」を用いることに相当する．



6.25 待ち行列のシミュレーション

サービス窓口が一つあって，

　客の到着する時間間隔は指数分布（母数 λ）に従う

　サービスの所要時間は正規分布 $N(\mu, \sigma^2)$ に従う

という場合の数値実験を行なって，待ち行列の状態を調べてみよう．

[**プログラムの設計**] 各種の方式があるが，ここでは初心者のために，記憶場所を十分に使って，なるべく明快に書くことにする．主な変数は

A_i	i 番目の客の到着時刻	X	到着間隔
T_j	j 番目の客の終了時刻	S	サービス所用時間
O	サービス開始時刻	Q	行列の長さ
i	「次に到着する客」の番号	j	「処理中の客」の番号

準備　前ページの方法により指数乱数 X を順に作って，到着時刻を

$$A_1 = 0 \qquad A_{i+1} = A_i + X \qquad (i = 1, 2, \cdots, n-1)$$

により計算して配列Aに入れ，「最初の客が到着した状態」

$$i = 1 \qquad j = 0 \qquad Q = 0$$

を初期値にして「到着」に進む．

到着時の処理　新 $i = i+1$ とし，

$$\begin{cases} Q = 0 \text{ ならば}, Q = 1, O = A_{i-1} \text{ にして「サービス開始」に進む．} \\ Q \geqq 1 \text{ ならば，新 } Q = Q+1 \text{ にして「時計」に進む．} \end{cases}$$

サービス開始時の処理　新 $j = j+1$ とし，正規乱数 S を作り，

$$T_j = O + S$$

により，サービス終了時点を算定して「時計」に進む．

サービス終了時の処理　今の客の待ち時間 $T_j - A_j$, 現時点における待ち行列の長さ Q などを出力し，Q から 1 を引き，

$$\begin{cases} Q = 0 \text{ ならば「到着」に進む．} \\ Q \geqq 1 \text{ ならば，} O = T_j \text{ として「サービス開始」に進む．} \end{cases}$$

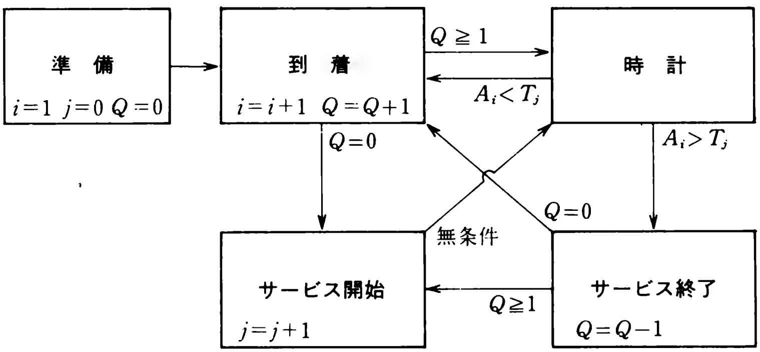
時計

$$\begin{cases} A_i \leqq T_j \text{ ならば「到着」に進む．} \\ A_i > T_j \text{ ならば「サービス終了」に進む．} \end{cases}$$

[プログラム例]

```
 10 DIM A(100),T(100)
 20 N=100
 30 PRINT "LAMBDA,MU,SIGMA=";
 40 INPUT L,C1,C2
 50 A(1)=0
 60 FŌR I=1 TŌ N-1
 70     X=-LŌG(RND)/L
 80     A(I+1)=A(I)+X
 90 NEXT I
100 I=1
110 J=0
120 Q=0
130 REM  *** TŌUTYAKU ***
140 I=I+1
150 IF I>N GŌTŌ 420
160 IF Q<=0 GŌTŌ 190
170 Q=Q+1
190 GŌTŌ 260                              時計へ
190 Q=1
200 Ō=A(I-1)
210 REM  *** SERVICE KAISI ***
220 J=J+1
230 GŌSUB 340                             正規乱数を作る
240 T(J)=Ō+S
250 REM    *** TŌKEI ***
260 IF A(I)<=T(J) GŌTŌ 140                到着へ
270 REM *** SERVICE SYURYŌU ***
280 PRINT A(J),T(J),T(J)-A(J),Q
290 Q=Q-1
300 IF Q<=0 GŌTŌ 140                      到着へ
310 Ō=T(J)
320 GŌTŌ 220                              サービス開始へ
```

```
330 REM   *** SEIKI RANSUU ***
340 Y=-6
350 FŌR K=1 TŌ 12
360     Y=Y+RND
370 NEXT K
380 S=C1+C2*Y                         S(C1, C2²) に換算
390 IF S>=0 GŌTŌ 410
400 RETURN
420 END
```




```
RUN
LAMBDA,MU,SIGMA
? 3,5,2
 0            3.51928       3.51928       11
 .175276      6.10783       5.93255       15
 .698696      9.03124       8.33255       20
 .89739       17.0419       16.1445       41
 1.04831      18.8557       17.8074       46
 1.80241      23.1985       21.3961       60
 1.87417      23.9396       22.0655       59
 2.92374      25.7708       22.847        64
 2.93948      29.5895       26.65         74
 3.28537      33.444        30.1586       86
```

6.26 演　　習

6.1 x の値を読み込み，

$$y = 3x^6+7x^4+5x^2+6$$

の値を計算し印刷するプログラムを作れ．

6.2 項数 n と実数 x の値を読み

$$x-\frac{x^3}{3}+\frac{x^5}{5}-\frac{x^7}{7}+\frac{x^9}{9}-\cdots+(-1)^{n-1}\frac{x^{2n-1}}{2n-1}$$

($\tan^{-1}x$ の展開式の第 n 項までの和) を計算するプログラムを作れ．

6.3 x および n の値を読み込み，次の級数の和を計算し印刷するプログラムを作れ．

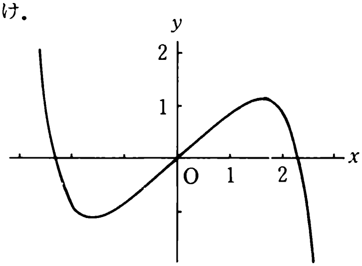
$$\sin x-\frac{\sin 2x}{2}+\cdots+(-1)^{n-1}\frac{\sin nx}{n}$$

6.4 **($f(x)=0$ と根)** 次の方程式を解け．

(i) $2x-\tan x = 0$

(ii) $2x-\cosh x = 0$

(iii) $x^2-\sin x = 0$



$y = 2x - \tan x$ のグラフ

［ヒント］ このような問題の場合，まず，グラフを描いて，根の箇数や概略の値を調べておくとよい．たとえば(i)の場合，グラフは右の図のようになるから，根は3箇あり，その一つは $x = 0$ で，他の二つは原点に関し対称であるから一方（たとえば正根）だけを求めればよく，その出発値としては $x = 2$（あるいは2.1，2.2など）を用いればよい，ということがわかる．

(ii)の $\cosh x$ は，周知のとおり，双曲線余弦 (hyperbolic cosine) で，定義は $\cosh x = (e^x+e^{-x})/2$，導関数は $(\cosh x)' = \sinh x = (e^x-e^{-x})/2$ である．

6.5 (3 次方程式，4 次方程式，5 次方程式)　次の方程式を解け．

(i)　$x^3-6x^2+11x-6=0$

(ii)　$x^4-x^3-x^2-x-2=0$

(iii)　$x^5+x^4-2x^3-2x^2+x+1=0$

6.6 (Kepler の方程式)

$$x-e\sin x=M$$

をニュートン法で解くプログラムを作れ．ただし離心率 e および平均近点離角 M はデーターとして与えられるものとする．

6.7 (三角関数表を作る)　よく知られている値

x	0°	30°	45°	60°	90°
$\sin x$	0	1/2	$\sqrt{2}/2$	$\sqrt{3}/2$	1

を補間して，三角関数の数表（0° から 90° まで 5° 間隔）を作り，厳密な値と比較せよ．

6.8　数値積分により，次の値を求めよ．

(i)　$\int_0^1 x^3dx$　　(ii)　$\int_0^1 e^x dx$　　(iii)　$\int_0^1 \frac{dx}{1+x^2}$

6.9 (対数表を作る)　周知のとおり，$1/x$ の不定積分は $\log x$ で，$\log 1=0$ であるから，1 から x まで $1/x$ を数値積分することによって $\log x$（の近似値）を求めることができるはずである．この方法により，$x=1$ から $x=2$ まで，$\Delta x=0.1$ 間隔の対数表を作成するプログラムを作れ．

［ヒント］　最初，$y(1)=0$ から出発して，

$$y(1.1)=y(1)+\int_{1.0}^{1.0}\frac{1}{x}dx$$

$$y(1.2)=y(1.1)+\int_{1.1}^{1.2}\frac{1}{x}dx$$

を計算して出力すればよい．積分公式は，まず台形公式か Simpson の公式のような簡単なもので試みて，精度があまり悪いようならば高次の公式に切り換える．なお，上記のようにして得られるのは自然対数である．常用対数表を作るには，上で得られた値を $\log_e 10$ で割ればよい．

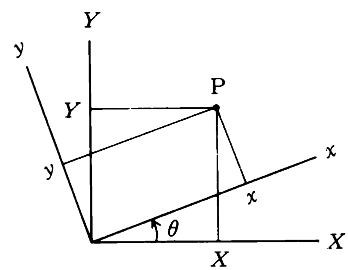
6.10 （無限区間の積分） 数値積分により$\int_0^\infty e^{x^2}dx$ の値を計算せよ.

［ヒント］ 無限区間の積分のためには，専用の公式（たとえばLaguerre–Gauss の公式など）もあるが，そういう特別な公式を使わなくても，被積分関数の値が十分に小さくなるまでとった有限区間の積分で近似しても，かなり良い精度が得られることが多い．いまの問題では，0 から 5 までを10等分して台形公式で積分する，という程度で十分の精度が得られる．

6.11 （座標変換） 右の図の X-Y 座標系で（X, Y）と表される点の x-y 座標（x, y）は次の式で求められる．

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} X \\ Y \end{pmatrix}$$

θ および X, Y を入力して x, y の値を計算して出力するプログラムを作れ．



6.12 （連乗） 行列 A, B とベクトル $\boldsymbol{x}$（行列は n 行 n 列，ベクトルは n 次元）を入力し，積

$$\boldsymbol{y} = AB\,\boldsymbol{x}$$

を計算して出力するプログラムを作れ．

［ヒント］ 先に積 AB を計算し，それを $\boldsymbol{x}$ に掛ける，という手順で計算すると計算量が多くなって損である．それよりも，先に $B\boldsymbol{x}$を計算し，それに A を掛ける方がよい．

6.13 （Fibonacci の数列） $a_0 = 0, a_1 = 1,$

$$a_{i+1} = a_i + a_{i-1} \qquad (i = 1, 2, \cdots)$$

で定義される数列（Fiboacci の数列という）を a_{20} まで計算して出力するプログラムを作れ．

6.14 (2項係数表) 2項係数($(a+b)^n$ の展開式の係数．その値は，n 箇のものから r 箇をとる組合せの数に等しい) ${}_nC_r$ は，よく知られている公式

$$ {}_nC_r = \frac{n!}{r!(n-r)!} $$

で計算することもできるが，階乗の計算は桁あふれを起こす危険があるから，漸化式

$$ \boxed{{}_nC_r = {}_{n-1}C_{r-1} + {}_{n-1}C_r} \quad (\text{ただし } {}_nC_0 = {}_nC_n = 1) $$

で計算する方がよい．この漸化式によって2項係数表($n=10$ まで)を計算して出力するプログラムを作れ．

6.15 (連立1次方程式) 次の連立1次方程式を解け．

(i) $\begin{cases} 36x+30y+24z = 3 \\ 30x+34y+26z = 4 \\ 24x+26y+21z = 4 \end{cases}$　　(ii) $\begin{cases} 6x+2y+3z = 5 \\ 18x+14y+11z = 21 \\ 24x+20y+16z = 30 \end{cases}$

6.16 (逆行列) 次の行列の逆行列を求めよ．

(i) $\begin{pmatrix} 2 & 4 & 6 \\ 3 & 8 & 7 \\ 5 & 7 & 21 \end{pmatrix}$　　(iii) $\begin{pmatrix} 68 & -41 & -17 & 10 \\ -41 & 25 & 10 & -6 \\ -17 & 10 & 5 & -3 \\ 10 & -6 & -3 & 2 \end{pmatrix}$

(iii) $\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 2 & 2 & 2 \\ 1 & 2 & 3 & 3 & 3 \\ 1 & 2 & 3 & 4 & 4 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix}$

6.17 (行列式) n 行 n 列の行列式の値を計算するプログラムを作れ．

[ヒント] Gauss の消去法の計算をして，そのピボットを全部掛け合わせればよい(ただし，行の交換をした場合は，交換のたびに符号を逆にする)．

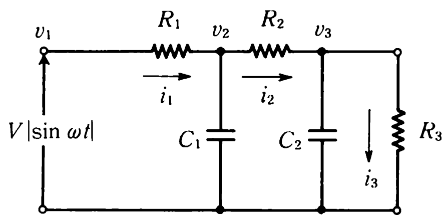
6.18 次の微分方程式の数値解を Euler 法で計算せよ．

（i） $y'+xy=0$　　$y(0)=1$

（ii） $yy'=x^2$　　$y(0)=1$

（iii） $y'+y=\sin x$　　$y(0)=1/2$

6.19 （***C*-*R* フィルター**）



右の図は，電源回路などに用いられる C-R 回路の一例である．左の端子に整流器から出てきた脈流を入れる．その電圧を

$$v_1 = V|\sin\omega t|$$

とする．端子電圧 v_1, v_2, v_3 の関係は，Ohm の法則により

$$i_1R_1 = v_1 - v_2$$

$$i_2R_2 = v_2 - v_3$$

$$i_3R_3 = v_3 - 0$$

となる．またコンデンサーの性質から

$$\frac{dv_2}{dt} = \frac{i_1 - i_2}{C_1} \qquad \frac{dv_3}{dt} = \frac{i_2 - i_3}{C_2}$$

以上をまとめると，次のような微分方程式が得られる．

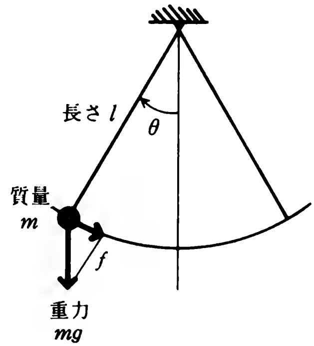
$$\begin{cases} \dfrac{dv_2}{dt} = \dfrac{1}{C_1}\left(\dfrac{v_1 - v_2}{R_1} - \dfrac{v_2 - v_3}{R_2}\right) \\ \dfrac{dv_3}{dt} = \dfrac{1}{C}\left(\dfrac{v_2 - v_3}{R_2} - \dfrac{v_3}{R_3}\right) \end{cases}$$

ただし，$v_1 = V|\sin\omega t|$

回路定数 R_1, R_2, C_1, C_2，負荷抵抗 R_3，電源電圧 V および角速度 ω，進み幅 Δt，計算打ち切り時刻 $t_{終}$ を読み込んで，上の微分方程式の数値解を計算するプログラムを作れ．初期条件は

$$t = 0 \text{ において } v_2 = v_3 = 0$$

とする．



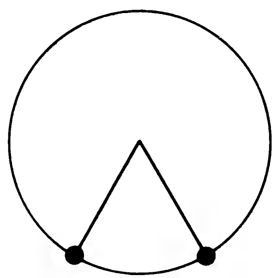
6.20 **（大振幅の振子）** 右図のような振子がある．おもりにかかる重力は mg（ただし g は重力加速度で約 980 cm/s²），その接線方向成分 f は，$f = mg\sin\theta$，したがって，トルクは $lf = lmg\sin\theta$，一方，質量慣性モーメントは ml であるから，運動方程式は

$$ml\frac{d^2s}{dt^2} = -lmg\sin\theta$$

ただし，s は接線にそって計った変位であり，θ と s の関係は（角度 θ の単位をラジアンとすれば）$s = l\theta$ である．この関係を入れると結局

$$\frac{d^2\theta}{dt^2} = -\frac{g}{l}\sin\theta$$

これが，大振幅の振子の微分方程式で，初等関数の範囲では解けない．$l = 30$ cm，初期条件 $\theta_0 = \pi/6$ ラジアン，初速 0 として，$t = 0$ から10秒間の解を求めよ（Euler 法ならば Δt を小さくとる必要がある．Runge–Kutta 法ならば Δt は 0.1秒ぐらいでよい）．これがうまく解けたら，さらに大きな θ_0 について計算してみるとよい（たとえば，$\theta_0 = (5/6)\pi$ ラジアン．右下の図参照）．



［注意］ θ_0 を大きくすると周期が長くなるから，解を求める範囲（終端時刻）をそれに合わせて拡大する（遅らせる）必要がある．一方，進み幅は小さくするわけにいかない（支点の直下を通過する時の速度が大きいから）．そのため，θ_0 を大きくすると計算所要時間が急増する．

6.27 文 献

各種の関数の近似式

1）山内, 宇野, 一松(編)；電子計算機のための数値計算法III, 培風館(1972)
2）一松 信；初等関数の数値計算, 教育出版（1974）
3）一松 信；近似式, 竹内書店（初版1963, 改訂版1972）
4）ヘイスティング（竹内訳）；電子計算機のための近似計算法, 東京図書（1973）
5）J.F. Hart ; *Computer approximation,* Wiley（1968）
6）Y.L. Luke ; *The special functions and their approximations,* I, II, Academic Press（1969）
7）磯田和男, 大野 豊(編)；数値計算ハンドブック, オーム社（1971）

補 間

8）ウェンドロフ（戸川訳）；理論数値解析, サイエンス社（1973）
9）森 正武；曲線と曲面, 教育出版（1974）

Gauss 型の積分公式の分点と係数の詳しい数表（3）はその他各種の公式についても詳しい）

10）山内, 宇野, 一松；電子計算機のための数値計算法III, 培風館(1972)
11）A.H. Stroud and D. Secrest ; *Gaussian quadrature formulas,* Prentice-Hall（1966）
12）デービス, ラビノウィッツ(森訳)；計算機のための数値積分法, 日本コンピュータ協会（1981）

微分方程式の数値解法

13）一松 信；微分方程式と解法, 教育出版（1977）
14）ヘンリチ（清水, 小林訳）；計算機による常微分方程式の解法, I, II, サイエンス社（1973）
15）C.W. Gear ; *Numercal initial value problems in ordinary differential equations,* Prentice-Hall（1981）

連立1次方程式の計算技法

16）戸川隼人；マトリクスの数値計算, オーム社（1971）
17）ウェストレイク（戸川訳）；線形計算ハンドブック, 培風館（1972）

7　データーの型，文字列処理

7.1　データーの型

BASIC で扱うことのできるデーターの種類には，大別して

- 数値データー
- 文字データー

がある．この内，数値データーの扱い方に関しては

整数型	（演算が速く，記憶場所が少なくて済む）
実数型	（普通の扱い方）
倍精度実数型	（高精度の計算ができる）

の区別がある．ただし，機種によっては，実数型だけしか使用できないものや，整数型だけしか使用できないものもある．

一方，文字データーに関しては，

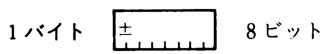
文字列型	（英数字の文字列を可変長の形で扱う）
漢字型	（漢字　〃　）

がある．正確には，それぞれ「英数字文字列型」「漢字文字列型」と呼ぶべきであろうが，現在のところ，後者を使用できる機種が少ないので，単に文字列型といえば前者を意味し，それに対し後者は漢字型と呼ばれている．

複素数型，多倍長整数型，多倍長実数型などは普通の BASIC では使用できない．論理型は正式の型としては扱われていないが，実質的にはその処理機能を持っている．大型コンピューターの BASIC には**MAT**文があり，いわば「ベクトル型」「マトリックス型」的な処理が可能であるが，パソコンの BASIC では使用できない．

7.2　データーの内部表現

パソコンで内部のデーターを扱う際の最小単位は8ビット（2進法の8桁）でこれを1バイト(byte)という．基本的な演算，記憶装置の出し入れ，周辺装置との間のデーターのやりとり，などはすべて（基本的には）1バイト単位で行なわれる*．



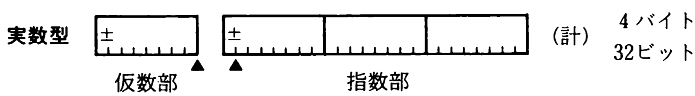
整数型のデーターは2バイト（16ビット）で表現される．16ビットの内の1ビットは符号を表すのに用いられる．また負数は補数**の形で表現される．そのため，整数型で扱うことのできる値の範囲は

$$-32768 \sim +32767$$

となる．（5万円というような大金は整数型では扱えないので注意！）．



実数型のデーターは，指数部（位どりを仮数部×$2^{(指数)}$の形で表す）1バイトと仮数部（有効数字を ± 0．******* の形で表す）3バイトの合計4バイトで表現され，処理される．



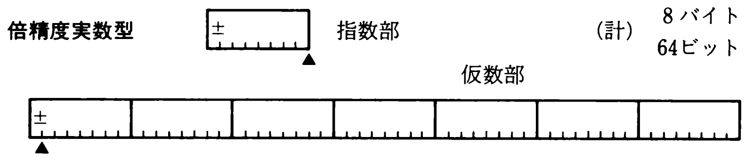
表現できる数値の範囲（絶対値）は $(1/2)\times 2^{-128} \fallingdotseq 10^{-39}$から，$(1-2^{-23})\times 2^{127} \fallingdotseq 10^{38}$まで，および0で，有効数字は2進法23桁であるから10進法換算約7桁である．

---

*　16ビット・パソコンの場合には，16ビット（2バイト）単位の扱いが多くなるが，それでもやはり1バイト単位で扱う処理がかなりある．

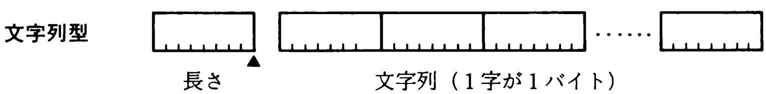
**　いわば「ゲタをはかせて正数になおした形」のこと．パソコンのBASICの整数型の場合，負数には $2^{16}=65536$を加えて表している．こうすると演算処理が簡単になるのである．

倍精度実数型のデーターは，指数部１バイト，仮数部７バイト，合計８バイトで扱われれる．



表現できる数値の範囲（絶対値）は約10^{-39}から約10^{38}まで，有効数字（仮数部の長さ）は２進法55桁であるから10進法換算約16桁である．

文字列型のデーターは，１文字を１バイトで表し，それに文字列の長さを表す１バイトを付加して，可変長（文字列の長さに合わせて処理する方式）の形で扱う．取扱える文字列の長さは最大255字である．



漢字は種類が多いので，１文字を表すのに２バイト必要である．



漢字型を使用できない機種では，これを普通，整数型データーとして扱うのであるが，漢字型を使用できるならば，これをさらに可変長データーとして扱うことができる．

1文字 2byte
↓
32000文字 ← 64 kbyte

7.3　属性文字と型宣言

変数の型の種別を指定する方法は

属性文字を付ける

型宣言による

の二通りある．前者の場合は，普通の変数名の後に次のような属性文字（型の種類を表す文字）を付ける．

整数型は	%	(例)	A%
実数型は	!	(例)	A!
倍精度実数型は	#	(例)	A#
文字列型は	$	(例)	A$
漢字型は	&	(例)	A&

後者の場合は，次のような型宣言文により，変数名の先頭の文字（頭文字）と型の対応関係を指定する．

整数型は	DEFINT	先頭の文字
実数型は	DEFSNG	〃
倍精度実数型は	DEFDBL	〃
文字列型は	DEFSTR	〃
漢字型は	(現在まだ使用できない)	

(例)「Iで始まる変数名はすべて整数型」ということを宣言するには

```
DEFINT I
```

と書けばよい．なお，次のような書き方もできる．たとえば

```
DEFDBL D,S,W
```

これは「DまたはSまたはWで始まる変数名はすべて倍精度実数型」ということを意味する．また

```
DEFSTR K-R
```

というような書き方もできる．これは「アルファベットのKからRまでの文字で始まる変数はすべて文字列型」ということを意味する．

> 特に型の指定がなければ，変数はすべて実数型として扱われる．型宣言と属性文字による指定が一致しない場合は属性文字が優先される．

数字定数の型を指定するには，数値の後に左記と同じ属性文字を付けるのが最も確実である．

(例)　5%　　整数型の5
　　　5！　　実数型の5
　　　5#　　倍精度実数型の5

数値定数は指数部を付けた形で書くことができる．書き方は

符号　数字の列，数字の列　E　符号　数字の列

または

符号　数字の列，数字の列　D　符号　数字の列

でそれらの

EまたはDより左が仮数部（いわば有効数字）

EまたはDより右が指数部

で，値

(仮数)×10$^{(指数)}$

を表す．このような形で書かれた定数の型は次のようになる．

Eを付ければ実数型

Dを付ければ倍精度実数型

属性文字も指数部も付いていない場合の扱いは，

小数点無しで－32768以上32767以下ならば整数型

小数点付きで7桁以下ならば実数型

小数点付きで8桁以上ならば倍精度実数型

としている機種が多いようである．

文字列型の定数は，文字列を2重引用符で囲んで表す．

(例)　"Ōh，wonderful！"

7.4　代入，配列，入出力

どの型についても，これまでに説明したのと同じ形で，代入をしたり，配列を用いたり，入出力を行なったりすることができる．

代　入　　正式の書き方は

LET　代入先の変数名＝式

であるがLETを省略して

代入先の変数名＝式

と書いてもよい．代入先の変数名の型と式の型が一致していれば全く問題ないが，一致していなくても，

整数型　　実数型　　倍精度実数型

の間では自動的に変換が行なわれる（まず，式の値を計算し，代入先の変数名の型に合わせて変換し，その上で代入を行なう）．整数型に変換する際，小数部は四捨五入されるのが普通である．

文字列変数の代入の場合，代入先の変数の文字列長さは，式の方で定まる文字列の長さに合わせて説定（必要ならば変更）される．

(例)　A$の内容がABCのとき，代入文

A$="PQRST"

を実行すると，以前の内容ABCは消され，かわりにPQRSTが入る．文字数は3字から5字になるが，このような代入も可能である．

配　列　　どの型についても，配列を用いることができる．宣言は

DIM　配列名（寸法）

により行なう．

(例)　DIM　A%(100),C#(30),M$(55)

入出力　倍精度実数型の出力は演算桁数に見合う十分な長さになる．

（例）　プログラム

```
10 A#=10000000000000000000
20 B#=3
30 C#=A#/B#
40 PRINT C#
50 END
```

の実行結果は次のようになる．

```
3.333333333333333D+17
```

文字列型データーの入力の場合，プログラムで字数を指定する必要はない．また，普通は，入力文字列を " 印で囲む必要もない．その場合，入力要求（？印）が出てから，入力データーの区切りを示す記号

コンマ　または　[改行]

の直前までが一つの文字列となる．ただし，" 印で囲んでおけば，コンマを含む文字列を入力することができる．

LINE INPUT 文　文字列型データーの入力に関しては，

LINE INPUT　文字列型変数名

という書き方ができる．このようにすると，1 行全部（[改行] まで）を一つの文字列として読み込んで文字列型変数として格納することができる．

7.5 文字列型の演算

文字列型データーに関し次のような処理が可能である．

1) **二つの文字列をつないで一つの文字列にする** これは＋印で表す．

(例) A$ の内容がCAN
B$ の内容がŌN
ならば，次のようになる．
A$+B$ はCANŌN

2) **左からn字目までをとり出す** これは次のように表す*． LEFT$ (文字列型変数名 , n)

(例) 人名が変数NAMAE$ に入っているとき，その頭文字をとり出すには，
LEFT$ (NAMAE$, 1)
とすればよい．

3) **右からn字目までをとり出す．** これは次のように表す． RIGHT$ (文字列型変数名 , n)

(例) I$ の内容が タノキン ならばRIGHT$ (I$, 2) の値は キン になる．

4) **文字列の中央部分（左からmの所からn字）をとり出す** これは次のように表す． MID$ (文字列変数名 , m , n)

(例) MŌJI$ の内容がCANDYのとき，
MID$ (MŌJI$, 2 , 3)
の値はANDになる．

---

* 一般に，文字列変数名のかわりに文字列式でもよい．以下同様．

5) **文字列の中に，指定された綴りが含まれているか否かを調べ，含まれている場合は，その位置（綴りの先頭が文字列の左から何字目にあるか）を調べる** これは，普通，

INSTR（文字列型変数名，"綴り"）

の形で書く*(これには＄が付かないことに注意!)．その値は

その文字列にその綴りが含まれない場合には0

含まれる場合はその位置（左からの字数で表した先頭位置）

になる．なお，照会開始位置を指定することもできる．その場合は次のように書く．

INSTR（照会開始位置，文字列型変数名，"綴り"）

（例） C＄の内容が

```
10 IF X=Y GŌTŌ 700
↑          ↑
└── 先頭    └──── 11字目
```

のとき，

INSTR（C＄，"GŌTŌ"）

の値は11になる．

6) **文字列の長さを調べる** これは

LEN（文字列型変数名）

と書く．

（例） S＄の内容が，シ゛ュケ゛ムシ゛ュケ゛ムコ゛コウノスリキレカイシ゛ャリスイキ゛ョノスイキ゛ョウマツウンキ゛ョウマツフウライマツクウネルトコロニスムトコロヤフ゛ラコウシ゛ノフ゛ラコウシ゛ハ゜イホ゜ハ゜イホ゜ハ゜イホ゜ノシューリンカ゛ンシューリンカ゛ンノク゛ーリンタ゛イク゛ーリンタ゛イノホ゜ンホ゜コヒ゜ーノホ゜ンホ゜コナチョウキュウメイノキュウスケ ならばLEN（S＄）の値は165になる．

---

* 一般には，INSTR（文字列式，文字列式）の形が許される．第1引数が被検索文字列，第2引数が検索する綴りを表す．

7.6　ASCII コード

パソコンの内部で，文字（漢字を除く）は0～225の番号で表されている．文字と番号の対応は，右の表のようになっている．この符号系は，英字，数字，記号*に関する限り，ASCII（アスキーと読む．アメリカの標準規格）の符号系と同一なので，ASCII コードと呼ばれている**．

ASCII コードを文字に変換するには次のように書く．***

```
CHR$(コード)
```

反対に，文字を ASCII コードに変換するには次のように書く．***

```
ASC(文字列型定数または変数名)
```

文字列の長さが2字以上の場合，先頭の1字のみが変換される．

（例）　Aの ASCII コードは65であるから，

```
PRINT CHR$(65)
```

を実行すれば文字Aが出力され

```
PRINT ASC("A")
```

を実行すれば数値65が出力される．

IF文によって文字列の大小を比較して分岐することができる．その際の大小関係は ASCII コードの大小に従う．

（例）　A＜B＜C＜…＜Z＜ア＜イ＜ウ＜…＜ン

比較する文字列は2字以上でもよい．その場合，次のように扱う．

先頭から順に比較する

文字列の終端より先は値ゼロとみなす

（例）　A＜AA＜AA空白＜AB

---

*　ただし￥および邦文用の記号（「　」など）を除く

**　正確には，「JIS C-6220 の8単位符号系」というのであるが，普通は ASCII と呼ばれている．

***　文字もコードも内部表現としては同じであるが BASIC における扱い方が異なる．（例）を参照のこと．

ASCII（JIS）コード表

文字	コード 10進	コード 16進	文字	コード 10進	コード 16進	文字	コード 10進	コード 16進	文字	コード 10進	コード 16進	文字	コード 10進	コード 16進
0	48	30	A	65	41	a	97	61	ア	177	B 1	ハ	202	C A
1	49	31	B	66	42	b	98	62	イ	178	B 2	ヒ	203	C B
2	50	32	C	67	43	c	99	63	ウ	179	B 3	フ	204	C C
3	51	33	D	68	44	d	100	64	エ	180	B 4	ヘ	205	C D
4	52	34	E	69	45	e	101	65	オ	181	B 5	ホ	206	C E
5	53	35	F	70	46	f	102	66	カ	182	B 6	マ	207	C F
6	54	36	G	71	47	g	103	67	キ	183	B 7	ミ	208	D 0
7	55	37	H	72	48	h	104	68	ク	184	B 8	ム	209	D 1
8	56	38	I	73	49	i	105	69	ケ	185	B 9	メ	210	D 2
9	57	39	J	74	4 A	j	106	6 A	コ	186	B A	モ	211	D 3
			K	75	4 B	k	107	6 B	サ	187	B B	ヤ	212	D 4
空白	32	20	L	76	4 C	l	108	6 C	シ	188	B C	ユ	213	D 5
!	33	21	M	77	4 D	m	109	6 D	ス	189	B D	ヨ	214	D 6
''	34	22	N	78	4 E	n	110	6 E	セ	190	B E	ラ	215	D 7
#	35	23	O	79	4 F	o	111	6 F	ソ	191	B F	リ	216	D 8
$	36	24	P	80	50	p	112	70	タ	192	C 0	ル	217	D 9
%	37	25	Q	81	51	q	113	71	チ	193	C 1	レ	218	D A
&	38	26	R	82	52	r	114	72	ツ	194	C 2	ロ	219	D B
'	39	27	S	83	53	s	115	73	テ	195	C 3	ワ	220	D C
(	40	28	T	84	54	t	116	74	ト	196	C 4	ン	221	D D
)	41	29	U	85	55	u	117	75	ナ	197	C 5			
*	42	2 A	V	86	56	v	118	76	ニ	198	C 6	ヲ	166	A 6
+	43	2 B	W	87	57	w	119	77	ヌ	199	C 7	ァ	167	A 7
,	44	2 C	X	88	58	x	120	78	ネ	200	C 8	ィ	168	A 8
−	45	2 D	Y	89	59	y	121	79	ノ	201	C 9	ゥ	169	A 9
	46	2 E	Z	90	5 A	z	122	80				ェ	170	A A
/	47	2 F							。		A 1	ォ	171	A B
	58	3 A	@	64	40	クリア	12	0 C	「		A 2	ャ	172	A C
;	59	3 B	[	91	5 B	改行	13	0 D	」		A 3	ョ	173	A D
<	60	3 C	¥	92	5 C	→	28	1 C	、		A 4	ッ	174	A E
=	61	3 D	]	93	5 D	←	29	1 D	・		A 5		175	A F
>	62	3 E	^	94	5 E	↑	30	1 E						
?	63	3 F	_	95	5 F	↓	31	1 F						

7.7 例　題

例題 1　商品名

いくつかの商品の，品名，単価，数量（売上げ数量）を入力し，各品目について（単価）×（数量）=（金額）の計算をして

品名　単価　数量　金額

の形式で作表し，最後に合計金額を出力するプログラムを作れ

［**解答**］ 品名は文字列変数H$で表し，単価，数量，金額はT，S，Kで表すことにする．品名欄が＊印ならば「データー終り」と解釈し，合計を出力することにする．作表の関係上，入力はREAD文で行なう（INPUT文で入力すると入力と出力が一緒になってうまくない）．

```
 10 G=0
 20 READ H$,T,S
 30 IF H$="*"GŌTŌ 80                    文字列の比較
 40 K=T*S
 50 PRINT H$,T,S,TAB(50);K
 60 G=G+K                 ↑———— 金額の印字位置を合計
 70 GŌTŌ 20                              欄と合わせるため．
 80 FŌR I=1 TŌ 11      ┐
 90     PRINT "------"  │ 横けいを引く．
100 NEXT I             ┘
110 PRINT "ごうけい",TAB(50);G
120 STŌP
130 DATA "リンゴ",200,3  ┐
140 DATA "ミカン",70,10  │ データーの例
150 DATA "*",0,0         ┘
```

［**出力例**］

```
RUN
ﾘﾝｺﾞ           200         3              600
ﾐｶﾝ            70          10             700
------------------------------------------------------------------
ｺﾞｳｹｲ                                     1300
Ok
```

例題 2　HAPPY BIRTHDAY

名前を入力し，それを HAPPY BIRTHDAY の歌に組み込んで，

HAPPY BIRTHDAY TŌ YŌU.
HAPPY BIRTHDAY TŌ YŌU.
HAPPY BIRTHDAY DEAR 名前 SAN.
HAPPY BIRTHDAY TŌ YŌU.

と出力するプログラムを作れ.

［**解答**］ このプログラムには，いろいろな書き方がある．下記はその一例であるが，これにこだわらずに各自で工夫するとよい．

```
 10 HB$="HAPPY BIRTHDAY "
 20 TY$="TŌ YŌU."
 30 PRINT "おなまえを いれてくだ゛さい"
 40 INPUT NAME$
 50 PRINT                              1 行あける
 60 PRINT HB$;TY$                     ｝二通りの書き方が可能
 70 PRINT HB$+TY$                     ｝どちらも結果は同じ
 80 PRINT HB$;"DEAR ";NAMAE$;" SAN."
 90 PRINT HB$;TY$
100 PRINT:PRINT:PRINT                  3 行あける
110 GŌTŌ 30
```

［**出力例**］

```
ｵﾅﾏｴｦ ｲﾚﾃｸﾀﾞｻｲ
? ｷﾖｼ

HAPPY BIRTHDAY TO YOU.
HAPPY BIRTHDAY TO YOU.
HAPPY BIRTHDAY DEAR ｷﾖｼ SAN.
HAPPY BIRTHDAY TO YOU.



ｵﾅﾏｴｦ ｲﾚﾃｸﾀﾞｻｲ
? ﾀｹｼ

HAPPY BIRTHDAY TO YOU.
HAPPY BIRTHDAY TO YOU.
HAPPY BIRTHDAY DEAR ﾀｹｼ SAN.
HAPPY BIRTHDAY TO YOU.
```

例題 3　　名簿の作成

クラスの名簿をフロッピー・ディスクまたはカセット・テープに入れておいて，必要に応じて，訂正したり，追加したり，削除したり，学生番号順に並べかえたり，アイウエオ順に並べかえたり，プリンターに出力したり，するためのプログラムを作れ．

［解答］ 簡単のため名簿は

　　　学生番号　　氏　名

の 2 項目だけとする(項目がもっと多くても要領は同じ)．ファイルの先頭には，タイトル（たとえば「**２ねん３くみのめいほ゛**」）と人数 N を記録し，それに続いて各人のデーター（各 2 項目）を順不同で入れることにする（別案としては，常に学生番号順またはアイウエオ順に整列した形でファイルを管理する方式も考えられる．実務にはその方が便利であるが，プログラムはやや複雑になる)．

アイウエオ順の並べかえは，ASCII コードの値の大小だけで処理することにする．じつは，そのようにすると，拗(よう)音や撥(はつ)音の文字（ャュョッ）のコードがアより前になるとか，濁点も 1 文字として扱われてしまうなど，多少，不自然なところもあるが，ここでは一応それでもよいということにしておく（まともに処理したければ，コード変換をすればよい)．整列の方法としては，最も簡単な交換法を用いることにする（これもプログラムを簡単にするためで，少人数ならば実用上問題ない)．

人数は200人以下とする．このぐらいの人数だと，ファイル全体が BASIC のデーター領域に入ってしまうので処理は簡単である（ファイルがメモリーに入りきらない場合には設計を全面的に変更する必要がある)．

以下に示すのは，ŌPEN 文およびプリンター出力の命令だが Level-3 用になっているが，そこを直せば他の機種でもだいたいそのまま使用できる（平片名を使用できない機種においては，そこを片仮名に直す)．

```
10 REM --------- めいほ゛フ゜ロ1 ---------
20 CLEAR 3000
30 DIM BANGOU(200),NAMAE$(200)
40 GOTO 930
50 REM --- しんき にゅうりょく ---
60 INPUT "タイトルを いれてくた゛さい";TITLE$
70 INPUT "せ゛んふ゛て゛ なんにんて゛すか";N
80 FOR I=1 TO N
90 INPUT "は゛んこ゛う";BANGOU(I)
100 INPUT "なまえ";NAMAE$(I)
110 NEXT I
120 RETURN
130 REM --- テ゛ィスフ゜レイ ---
140 PRINT "▒▒▒▒▒▒▒▒▒▒";TITLE$;"▒▒▒▒▒▒▒▒▒▒"
150 FOR I=1 TO N
160 PRINT USING"### ";I;
170 PRINT USING"########## ";BANGOU(I);
180 PRINT NAMAE$(I)
190 NEXT I
200 INPUT "OK?",OK$
210 RETURN
220 REM --- フ゜リンターに しゅつりょく ---
230 OPEN "O",#1,"LPTO:"
240 PRINT #1
250 PRINT #1,"▒▒▒▒▒▒▒▒▒▒";TITLE$;"▒▒▒▒▒▒▒▒▒▒"
260 PRINT #1
270 FOR I=1 TO N
280 PRINT #1,USING"### ";I;
290 PRINT #1,USING"##########";BANGOU(I);
300 PRINT #1,"    ";NAMAE$(I)
310 NEXT I
320 CLOSE #1
330 RETURN
340 REM --- めいほ゛の ていせい ---
350 INPUT "き゛ょうは゛んこ゛う";I
360 INPUT "は゛んこ゛う";BANGOU(I)
370 INPUT "なまえ";NAMAE$(I)
380 RETURN
390 REM --- めいほ゛の ついか ---
400 N=N+1
410 INPUT "は゛んこ゛う";BANGOU(N)
420 INPUT "なまえ";NAMAE$(N)
430 INPUT "これて゛おわりて゛すか?(Y/N)",FIN$
440 IF FIN$="N" GOTO 400
450 RETURN
```

```
460 REM --- じしょ から さくじょ ---
470 INPUT "ぎょうばんごう";I
480 PRINT ENGLISH$(I),JAPANESE$(I)
490 INPUT "OK?(Y/N)";OK$
500 IF OK$="N" GOTO 470
510 IF I=N GOTO 560
520 FOR J=I TO N-1
530 ENGLISH$(J)=ENGLISH$(J+1)
540 JAPANESE$(J)=JAPANESE$(J+1)
550 NEXT J
560 N=N-1
570 RETURN
580 REM --- ディスクに かきこむ ---
590 OPEN "O",#1,"えいわじてん"
610 PRINT #1,N
620 FOR I=1 TO N
630 PRINT #1,ENGLISH$(I);",";JAPANESE$(I)
640 NEXT I
650 CLOSE #1
660 RETURN
670 REM --- ディスクから よみこむ ---
680 OPEN "I",#1,"えいわじてん"
690 INPUT #1,N
700 FOR I=1 TO N
710 INPUT #1,ENGLISH$(I),JAPANESE$(I)
720 NEXT I
730 CLOSE #1
740 RETURN
750 REM --- ばんごうじゅんに ならべかえる ---
760 FOR M=N TO 2 STEP -1
770 FOR I=2 TO M
780 IF ENGLISH$(I-1)<ENGLISH$(I) GOTO 810
790 SWAP ENGLISH$(I-1),ENGLISH$(I)
800 SWAP JAPANESE$(I-1),JAPANESE$(I)
810 NEXT I
820 NEXT M
830 RETURN
840 REM --- アイウエオじゅんに ならべかえる ---
850 FOR M=N TO 2 STEP -1
860 FOR I=2 TO M
870 IF JAPANESE$(I-1)<JAPANESE$(I) GOTO 900
880 SWAP ENGLISH$(I-1),ENGLISH$(I)
890 SWAP JAPANESE$(I-1),JAPANESE$(I)
900 NEXT I
910 NEXT M
920 RETURN
```

```
930 REM --- switching center ---
940 CLS
950 PRINT "これは めいぼ  がんり の プログラム です。"
960 PRINT
970 PRINT "しんきさくせい・・・・・・・・・・・・・・・・・・・・1"
980 PRINT "プリンターにしゅつりょく・・・・・・・・・・・・・・2"
990 PRINT "ていせい・・・・・・・・・・・・・・・・・・・・・・3"
1000 PRINT "ついか・・・・・・・・・・・・・・・・・・・・・・・4"
1010 PRINT "さくじょ・・・・・・・・・・・・・・・・・・・・・・5"
1020 PRINT "ディスクにかきこむ・・・・・・・・・・・・・・・・6"
1030 PRINT "ディスクからよみこむ・・・・・・・・・・・・・・・7"
1040 PRINT "ばんごうじゅんに ならべかえる・・・・・・・8"
1050 PRINT "アイウエオ じゅんに ならべかえる・・・・・・・9"
1060 PRINT "ディスプレイ・・・・・・・・・・・・・・・・・10"
1070 PRINT
1080 INPUT "------------->>> どれにしますか";K
1090 CLS
1100 ON K GOSUB 50,220,340,390,460,580,670,750,840,130
1110 GOTO 940
```

[出力例]

░░░░░░ゼミの めいぼ ░░░░░░

1	9146	よねばやし せつこ
2	9034	がとう まさひこ
3	9043	きむら まり
4	9131	むらなか のりこ
5	9036	かすが ようこ
6	9105	はせがわ やすひろ
7	9111	ひらやま よう
8	9003	あさなぎ としお
9	9007	いしぐろ りゅうじ
10	9095	ながやま のぶひこ
11	9140	やすまつ めぐみ
12	9016	おがわ としひこ

[アイウエオ順に並べかえた結果]

░░░░░░ゼミの めいぼ ░░░░░░

1	9003	あさなぎ としお
2	9007	いしぐろ りゅうじ
3	9016	おがわ としひこ
4	9036	かすが ようこ
5	9034	がとう まさひこ
6	9043	きむら まり
7	9095	ながやま のぶひこ
8	9105	はせがわ やすひろ
9	9111	ひらやま よう
10	9131	むらなか のりこ
11	9140	やすまつ めぐみ
12	9146	よねばやし せつこ

例題 4　英和辞典

簡単な英和辞典（単語一つに対し訳語は一つだけとする）をフロッピー・ディスクまたはカセット・テープに入れておいて，必要に応じて，訂正したり，追加したり，削除したり，見出し語の ABC 順に並べかえたり，プリンターに出力したりするためのプログラムを作れ。

[解答]　辞書の規模はメモリー (BASIC のデーター領域) に収容できる程度とし，次のような配列の形で扱うことにする。

添字	ENGLISH$	JAPANESE$
1	CAT	ネコ
2	DŌG	イヌ
3	LIŌN	ライオン
⋮	⋮	⋮

処理の要領は例題 3 とだいたい同じなので，例題 3 のプログラムの変数名を

BANGŌU ⟶ ENGLISH$
NAMAE$ ⟶ JAPANESE$

に変更し，ファイル名を適当に（たとえば

MEIBŌ ⟶ えいわしﾞてん

に）変更し，見出しを

はﾞんこﾞう ⟶ えいこﾞ
なまえ ⟶ にほんこﾞ

のようなぐあいに変更すれば，英和辞典のプログラムになる。このような書きかえには，後述（例題 7）の TŌŌL 1 を用いて行なうとよい。行20の CLEAR 文は文字列型変数のための記憶領域の広さを設定するもので，このプログラムではデーターの大部分が文字列型であるから，大きめに 8000(8KB) とした。

```
10 REM --------'えいわ し゛てん ---------
20 CLEAR 8000
30 DIM ENGLISH$(500),JAPANESE$(500)
40 GOTO 930
50 REM --- しんき にゅうりょく ---
60 TITLE$="ENGLISH-JAPANESE DICTIONARY"
70 N=1
80 REM --- loop ---
90 INPUT "えいこ゛";ENGLISH$(N)
100 INPUT "にほんこ゛";JAPANESE$(N)
110 IF ENGLISH$(N)="" THEN N=N-1:RETURN
120 N=N+1 : GOTO 90
130 REM --- テ゛ィスフ゜レイ ---
140 PRINT "▒▒▒▒▒▒▒▒";TITLE$;"▒▒▒▒▒▒▒▒"
150 FOR I=1 TO N
160 PRINT USING"### ";I;
170 PRINT ENGLISH$(I),
180 PRINT JAPANESE$(I)
190 NEXT I
200 INPUT "OK?",OK$
210 RETURN
220 REM --- フ゜リンターに しゅつりょく ---
230 OPEN "O",#1,"LPTO:"
240 PRINT #1
250 PRINT #1,"▒▒▒▒▒▒▒▒";TITLE$;"▒▒▒▒▒▒▒▒"
260 PRINT #1
270 FOR I=1 TO N
280 PRINT #1,USING"### ";I;
290 PRINT #1,ENGLISH$(I),
300 PRINT #1,JAPANESE$(I)
310 NEXT I
320 CLOSE #1
330 RETURN
340 REM --- し゛しょの ていせい ---
350 INPUT "き゛ょうは゜んこ゛う";I
360 INPUT "えいこ゛";ENGLISH$(I)
370 INPUT "にほんこ゛";JAPANESE$(I)
380 RETURN
390 REM --- たんこ゛の ついか ---
400 N=N+1
410 INPUT "えいこ゛";ENGLISH$(N)
420 INPUT "にほんこ゛";JAPANESE$(N)
430 INPUT "これて゛おわりて゛すか?(Y/N)",FIN$
440 IF FIN$="N" GOTO 400
450 RETURN
```

```
460 REM --- めいほ゛ から さくし゛ょ ---
470 INPUT "き゛ょうは゛んこ゛う";I
480 PRINT BANGOU(I),NAMAE$(I)
490 INPUT "OK?(Y/N)";OK$
500 IF OK$="N" GOTO 470
510 IF I=N GOTO 560
520 FOR J=I TO N-1
530 BANGOU(J)=BANGOU(J+1)
540 NAMAE$(J)=NAMAE$(J+1)
550 NEXT J
560 N=N-1
570 RETURN
580 REM --- テ゛ィスクに かきこむ ---
590 OPEN "O",#1,"MEIBO"
600 PRINT #1,TITLE$
610 PRINT #1,N
620 FOR I=1 TO N
630 PRINT #1,BANGOU(I),NAMAE$(I)
640 NEXT I
650 CLOSE #1
660 RETURN
670 REM --- テ゛ィスクから よみこむ ---
680 OPEN "I",#1,"MEIBO"
690 INPUT #1,TITLE$,N
700 FOR I=1 TO N
710 INPUT #1,BANGOU(I),NAMAE$(I)
720 NEXT I
730 CLOSE #1
740 RETURN
750 REM --- は゛んこ゛うし゛ゅんに ならへ゛かえる ---
760 FOR M=N TO 2 STEP -1
770 FOR I=2 TO M
780 IF BANGOU(I-1)<BANGOU(I) GOTO 810
790 SWAP BANGOU(I-1),BANGOU(I)
800 SWAP NAMAE$(I-1),NAMAE$(I)
810 NEXT I
820 NEXT M
830 RETURN
840 REM --- アイウエオし゛ゅんに ならへ゛かえる ---
850 FOR M=N TO 2 STEP -1
860 FOR I=2 TO M
870 IF NAMAE$(I-1)<NAMAE$(I) GOTO 900
880 SWAP BANGOU(I-1),BANGOU(I)
890 SWAP NAMAE$(I-1),NAMAE$(I)
900 NEXT I
910 NEXT M
920 RETURN
```

```
930 REM --- switching center ---
940 CLS
950 PRINT "これは えいわじてん へんしゅう の プログラム です。"
960 PRINT
970 PRINT "しんきさくせい.....................1"
980 PRINT "プリンターにしゅつりょく..............2"
990 PRINT "ていせい.........................3"
1000 PRINT "ついか..........................4"
1010 PRINT "さくじょ........................5"
1020 PRINT "ディスクにかきこむ.................6"
1030 PRINT "ディスクからよみこむ................7"
1040 PRINT "ABC じゅんに ならべかえる..........8"
1050 PRINT "アイウエオ じゅんに ならべかえる.......9"
1060 PRINT "ディスプレイ.....................10"
1070 PRINT
1080 INPUT "-------------->>> どれにしますか";K
1090 CLS
1100 ON K GOSUB 50,220,340,390,460,580,670,750,840,130
1110 GOTO 940
```

[出力例]

```
░░░░░░░░ENGLISH-JAPANESE DICTIONARY░░░░░░░░

   1 CAT        ネコ
   2 DOG        イヌ
   3 LION       ライオン
   4 TIGER      トラ
   5 RAT        ネズミ
   6 WOLF       オオカミ
   7 CAMEL      ラクダ
```

[**ABC 順**のリスト]

```
░░░░░░░░ENGLISH-JAPANESE DICTIONARY░░░░░░░░

   1 CAMEL      ラクダ
   2 CAT        ネコ
   3 DOG        イヌ
   4 LION       ライオン
   5 RAT        ネズミ
   6 TIGER      トラ
   7 WOLF       オオカミ
```

[アイウエオ順のリスト]

```
░░░░░░░░ENGLISH-JAPANESE DICTIONARY░░░░░░░░

   1 DOG        イヌ
   2 WOLF       オオカミ
   3 TIGER      トラ
   4 CAT        ネコ
   5 RAT        ネズミ
   6 LION       ライオン
   7 CAMEL      ラクダ
```

例題 5　子供銀行の預金業務

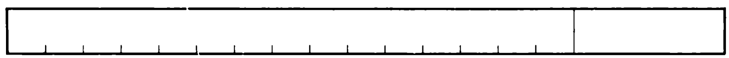
ディスクをランダム・アクセス*・ファイルの形で使う練習に，子供銀行の預金業務を処理するプログラムを作れ．

［**解答**］ 処理としては

新規加入　　解　約　　預け入れ　　払い戻し

を扱うものとし，必要に応じて預金残高リストを作成できるようにする．レコードの形式は

加入者氏名　　　　残　高



（15バイト）　　　（4バイト）

とする．文法の詳細は付録A.3を参照されたい．行番号6000～のサブルーチンは，処理内容，口座番号，加入者氏名，金額，残高を1行にプリントするための工夫である．

［**プログラム例**］

```
10 REM --- こど゛も ぎ゛んこう ---
20 PRINT "しんき かにゅう…………………………1"
30 PRINT "かいやく……………………………………2"
40 PRINT "あず゛けいれ………………………………3"
50 PRINT "はらいもど゛し……………………………4"
60 PRINT "リスト………………………………………5"
70 PRINT
80 INPUT K
90 ON K GOSUB 1000,2000,3000,4000,5000
100 GOTO 10
1000 OPEN "R",#1,"BANK"
1010 INPUT "なまえ";NAMAE$
1020 INPUT "きんが゛く";KINGAKU : ZANDAKA=KINGAKU
1030 FIELD #1,30 AS N$,4 AS Z$
1040 LSET N$=NAMAE$
1050 LSET Z$=MKS$(KINGAKU)
1060 BANGOU=LOF(1)+1
1070 PUT #1,BANGOU
1080 CLOSE #1
1090 MIDASI$="しんきかにゅう"
1100 GOSUB 6000
1110 RETURN
```

*　234ページ参照

```
2000 OPEN "R",#1,"BANK"
2010 INPUT "こうさ゛は゛んこ゛う";BANGOU
2020 FIELD #1,30 AS N$,4 AS Z$
2030 GET #1,BANGOU
2040 NAMAE$=N$
2050 ZANDAKA=CVS(Z$)
2060 LSET N$=""
2070 LSET Z$=MKS$(0)
2080 PUT #1,BANGOU
2090 CLOSE #1
2100 MIDASI$="かいやく"
2110 GOSUB 6000
2120 RETURN
3000 OPEN "R",#1,"BANK"
3010 INPUT "こうさ゛は゛んこ゛う";BANGOU
3020 INPUT "きんか゛く";KINGAKU
3030 FIELD #1,30 AS N$,4 AS Z$
3040 GET #1,BANGOU
3050 NAMAE$=N$
3060 ZANDAKA=CVS(Z$)+KINGAKU
3070 LSET Z$=MKS$(ZANDAKA)
3080 PUT #1,BANGOU
3090 CLOSE #1
3100 MIDASI$="あす゛けいれ"
3110 GOSUB 6000
3120 RETURN
4000 OPEN "R",#1,"BANK"
4010 INPUT "こうさ゛は゛んこ゛う";BANGOU
4020 INPUT "きんか゛く";KINGAKU
4030 FIELD #1,30 AS N$,4 AS Z$
4040 GET #1,BANGOU
4050 NAMAE$=N$
4060 ZANDAKA=CVS(Z$)-KINGAKU
4070 IF ZANDAKA<0 GOTO 4150
4080 LSET Z$=MKS$(ZANDAKA)
4090 PUT #1,BANGOU
4100 CLOSE #1
4110 MIDASI$="はらいもと゛し"
4120 GOSUB 6000
4130 RETURN
4140 PRINT "さ゛んた゛か ふそく"
4150 PRINT "け゛んさ゛い さ゛んた゛か",CVS(Z$)
4160 CLOSE #1
4170 RETURN
```

```
5000 OPEN "R",#1,"BANK"
5010 OPEN "O",#2,"LPTO:"
5020 PRINT #2 : PRINT #2
5030 PRINT #2,"こうさ゛は゛んこ゛う","かにゅうしゃ しめい","さ゛んた゛か"
5040 FIELD #1,30 AS N$,4 AS Z$
5050 BMAX=LOF(1)
5060 FOR BANGOU=1 TO BMAX
5070 GET #1,BANGOU
5080 NAMAE$=N$
5090 ZANDAKA=CVS(Z$)
5100 PRINT #2,BANGOU,
5110 PRINT #2,USING"&                    &";NAMAE$;
5120 PRINT #2,ZANDAKA
5130 NEXT BANGOU
5140 CLOSE #1
5150 CLOSE #2
5160 RETURN
6000 OPEN "O",#2,"LPTO:"
6010 PRINT #2,USING"&              &";MIDASI$;
6020 PRINT #2,USING" ####### ";BANGOU;
6030 PRINT #2,USING"&              &";NAMAE$;
6040 PRINT #2,USING" ####### ";KINGAKU;
6050 PRINT #2,USING" ####### ";ZANDAKA
6060 CLOSE #2
6070 RETURN
```

[出力例 —— 経過記録]

```
しんきかにゅう            2 すき゛もと ひろし       1800      1800
しんきかにゅう            3 はたふ゛  よしたか       950       950
しんきかにゅう            4 なかやま のふ゛ひこ     2000      2000
しんきかにゅう            5 ふし゛い ひて゛ゆき     1300      1300
あす゛けいれ              2 すき゛もと ひろし        500      2300
はらいもと゛し            5 ふし゛い ひて゛ゆき      120      1180
```

[出力例 —— 残高リスト]

```
こうさ゛は゛んこ゛う     かにゅうしゃ しめい     さ゛んた゛か
  1                     おか゛わ はるの           3000
  2                     すき゛もと ひろし         2300
  3                     はたふ゛  よしたか        950
  4                     なかやま のふ゛ひこ       2000
  5                     ふし゛い ひて゛ゆき       1180
```

例題 6　在庫管理

前問のプログラムを改造して在庫管理のプログラムを作れ．

[出題の意図]　前問のテーマ「子供銀行の預金業務」は，実務にはほど遠い，単なる遊びのように思われるかもしれないが，それをほんの少し改造すると在庫管理のプログラムを作ることができる．在庫管理は，

商　品　　部分品　　材　料　　消耗品

などの手持ち量を監視して，在庫切れを起こさないように（しかし，手持ちを多くしすぎて無駄を生ずることのないように）調節する仕事で，商店や工場はもちろんのこと，研究室や計算センターなど，あらゆる活動に役立つものであるから，この機会に習得しておくとよい．要領は，前問の預金業務とだいたい同じで，

「預け入れ」を「入庫量」に
「払い戻し」を「出庫量」に
「残　高」　を「在庫量」に

書きかえれば，在庫管理プログラムの主部分ができあがるから，あと，適当に手を加えればよい．

[プログラム例]　このような書きかえには，後述(例題７)のTŌŌL　1を用いるとよい．そういう方法で作ったプログラムの一例を以下に示す（じつは，このプログラムのもとになった「子供銀行プログラム」は旧版で，その後，子供銀行プログラムに少し手を入れているので，現在の版を機械的に書きかえた場合にはこれと少し違ったものになるであろう）．

```
10 REM --- ざいこ かんり ---
20 PRINT "ひんめい ついか・・・・・・・・・・・・・・・・・・・・1"
30 PRINT "ひんめい さくじょ・・・・・・・・・・・・・・・・・・2"
40 PRINT "にゅうこ・・・・・・・・・・・・・・・・・・・・・・・・3"
50 PRINT "しゅっこ・・・・・・・・・・・・・・・・・・・・・・・・4"
60 PRINT "リスト・・・・・・・・・・・・・・・・・・・・・・・・・5"
70 PRINT
80 INPUT K
90 ON K GOSUB 1000,2000,3000,4000,5000
100 GOTO 10
```

```
1000 OPEN "R",#1,"ざいこファイル"
1010 INPUT "ひんめい",HINMEI$
1020 INPUT "すうりょう",SUURYOU
1030 FIELD #1,30 AS N$,4 AS Z$
1040 LSET N$=HINMEI$
1050 LSET Z$=MKS$(SUURYOU)
1060 BANGOU=LOF(1)+1
1070 PUT #1,BANGOU
1080 CLOSE #1
1090 OPEN "O",#2,"LPTO:"
1100 MIDASI$="しんき とうろく"
1110 GOSUB 6000
1120 CLOSE #2
1130 RETURN
2000 OPEN "R",#1,"ざいこファイル"
2010 INPUT "コード",BANGOU
2020 FIELD #1,30 AS N$,4 AS Z$
2030 GET #1,BANGOU
2040 HINMEI$=N$
2050 ZAIKO=CVS(Z$)
2060 LSET N$=""
2070 LSET Z$=MKS$(0)
2080 PUT #1,BANGOU
2090 CLOSE #1
2100 OPEN "O",#2,"LPTO:"
2110 MIDASI$="ひんめい さくじょ"
2120 GOSUB 6000
2130 CLOSE #2
2140 RETURN
3000 OPEN "R",#1,"ざいこファイル"
3010 INPUT "コード",BANGOU
3020 INPUT "すうりょう",SUURYOU
3030 FIELD #1,30 AS N$,4 AS Z$
3040 GET #1,BANGOU
3050 HINMEI$=N$
3060 ZAIKO=CVS(Z$)+SUURYOU
3070 LSET Z$=MKS$(ZAIKO)
3080 PUT #1,BANGOU
3090 CLOSE #1
3100 OPEN "O",#2,"LPTO:"
3110 MIDASI$="にゅうこ"
3120 GOSUB 6000
3130 CLOSE #2
3140 RETURN
```

```
4000 OPEN "R",#1,"ざいこファイル"
4010 INPUT "コード",BANGOU
4020 INPUT "すうりょう",SUURYOU
4030 FIELD #1,30 AS N$,4 AS Z$
4040 GET #1,BANGOU
4050 HINMEI$=N$
4060 ZAIKO=CVS(Z$)-SUURYOU
4070 IF ZAIKO<0 GOTO 4150
4080 LSET Z$=MKS$(ZAIKO)
4090 PUT #1,BANGOU
4100 CLOSE #1
4110 OPEN "O",#2,"LPT0:"
4120 MIDASI$="しゅっこ"
4130 GOSUB 6000
4140 CLOSE #2
4150 RETURN
4160 PRINT "ふそく"
4170 PRINT "げんざいりょう",CVS(Z$)
4180 CLOSE #1
4190 RETURN
5000 OPEN "R",#1,"ざいこファイル"
5010 OPEN "O",#2,"LPT0:"
5020 PRINT #2 : PRINT #2
5030 PRINT #2,"コード","ひんめい","ざいこりょう"
5040 FIELD #1,30 AS N$,4 AS Z$
5050 BMAX=LOF(1)
5060 FOR BANGOU=1 TO BMAX
5070 GET #1,BANGOU
5080 HINMEI$=N$
5090 ZAIKO=CVS(Z$)
5100 PRINT #2,BANGOU,HINMEI$,ZAIKO
5110 NEXT BANGOU
5120 CLOSE #1
5130 CLOSE #2
5140 RETURN
6000 PRINT #2,USING"&          &";MIDASI$;
6010 PRINT #2,USING" #######";BANGOU;
6020 PRINT #2,USING"&          &";HINMEI$;
6030 PRINT #2,USING" #######";SUURYOU,ZAIKO
6040 RETURN
```

例題 7 ―― 変数名の付けかえ

ASCII セーブ* されたプログラムはデーターとして読み込むことができる．これを用いて変数名の付けかえを自動的に行なうプログラムを作れ．

[解答] 行100〜230は，プログラムを文字列型配列A$ に読み込むプログラムである．LINE INPUT の場合，IF EŌF という終端検出法を使えないので，ŌN ERRŌR で終端を検出している．行210と220は，error の原因が end of file であるか否かを調べている（このエラー番号は Level-3 用である．他機種の場合はここを書きかえる必要があるが，めんどうならば2行削除してもよい）．該当変数名の位置は関数INSTRで調べ，それより左側の部分を関数LEFT$ でとり出してALに入れ，右側の部分を関数RIGHT$ でとり出してARに入れ，ALと新変数名とARをつないで新しい文を作っている．行390〜440は，該当文字列の左右が英数字以外であるか否かのチェックである．

```
10 REM ▒▒▒▒▒▒▒▒ TOOL1 ▒▒▒▒▒▒▒▒
20 REM ▒                      ▒
30 REM ▒   へんすうめい の つけかえ   ▒
40 REM ▒                      ▒
50 REM ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒
60 CLEAR 8000
70 DIM A$(300)
80 OPEN "O",#2,"LPTO:"
90 REM ▒▒▒▒▒▒▒▒ ﾌﾟﾛｸﾞﾗﾑの よみこみ ▒▒▒▒▒▒▒▒
100 INPUT "ﾌｧｲﾙめいは?";F$
110 OPEN "I",#1,F$
120 FOR L=1 TO 400
130 LINE INPUT #1,A$(L)
140 PRINT A$(L)
150 ON ERROR GOTO 200
160 NEXT L
170 PRINT "きﾞょうすうｵｰﾊﾞｰ"
180 CLOSE #1
190 END
```

---

* SAVE "ファイル名",A という指定でカセット・テープまたはフロッピー・ディスクに記録すること．231ページ参照．

```
200 CLOSE #1
210 IF ERL<>130 THEN PRINT"ERL=";ERL:END
220 IF ERR<>54 THEN PRINT"ERR=";ERR:END
230 REM --------- end of file ---------
240 N=L-1
250 REM ▒▒▒▒▒▒▒ へんすうめい の つけかえ ▒▒▒▒▒▒▒
260 PRINT #2,"---- へんすうめい の つけかえ ----"
270 INPUT "つけかえる へんすうめいは?";H$
280 M=LEN(H$)
290 IF M=0 GOTO 540
300 INPUT "あたらしい へんすうめいは?";T$
310 MT=LEN(T$)
320 PRINT #2,"つけかえる へんすうめい";H$
330 PRINT #2,"あたらしい へんすうめい";T$
340 FOR L=1 TO N
350 I=1
360 J=INSTR(I,A$(L),H$)
370 IF J=0 GOTO 520
380 LC=ASC(MID$(A$(L),J-1,1))
390 IF (&H41<=LC)AND(LC<=&H5A) GOTO 500
400 IF (&H30<=LC)AND(LC<=&H39) GOTO 500
410 IF J+M>LEN(A$(L)) GOTO 450
420 RC=ASC(MID$(A$(L),J+M,1))
430 IF (&H41<=RC)AND(RC<=&H5A) GOTO 500
440 IF (&H30<=RC)AND(RC<=&H39) GOTO 500
450 PRINT #2,A$(L)
460 AL$=LEFT$(A$(L),J-1)
470 AR$=RIGHT$(A$(L),LEN(A$(L))-J+1-M)
480 A$(L)=AL$+T$+AR$
490 PRINT #2,A$(L)
500 I=J+MT
510 GOTO 360
520 NEXT L
530 GOTO 270
540 REM --- あたらしいﾌﾟﾛｸﾞﾗﾑをいんさつ ---
550 PRINT #2
560 FOR L=1 TO N
570 PRINT #2,A$(L)
580 NEXT L
590 REM ▒▒▒▒▒▒▒ ﾃﾞｨｽｸに かきこむ ▒▒▒▒▒▒▒
600 INPUT "ﾌｧｲﾙめいは?",G$
610 OPEN "O",#1,"1:"+G$
620 FOR L=1 TO N
630 PRINT #1,A$(L)
640 NEXT L
650 CLOSE #1
660 CLOSE #2
670 END
```

例題 8　インデンテーション

BASIC のプログラムは，メモリーを節約するため，普通，左づめの形で記憶され，表示されるが，できれば，くりかえし部分に段落を付けて（右のページの例のように）表示，印刷する方が見易い．このためのプログラムを作れ．

［解答］　関数 INSTR で綴り FŌR と NEXT を検出し，FŌR で 2 字下げて NEXT で 2 字もどすようにしたのが以下のプログラムである．

行100〜180は，ASCII セーブされたプログラムを配列 A$ に読み込むプログラムで，処理の主部分（行250以後）を明快にするためにこのようにしたが，別案としては，1 行読み込んですぐに処理してプリントする，という方法も考えられる（そのようにすれば，いくら長いプログラムでも処理できて便利である）．

行150の　ŌN　ERRŌR　GŌTŌ　は，end of file を検出するための便法で，プログラムの最後まで読み終るとERRŌRということになって行200に進む．そのとき，行数のカウンターLは一つ進んでいるから，行240で1を引いて，正しい行数をNに入れる．

変数LEVELはFŌR〜NEXT文の入れ子の深さ（何重か）を示すもので，最初0で出発し，綴りFŌRを見つけたらLEVELに1を加え，NEXTを見つけたら1を引く．このプログラムは構文チェックをしていないから，注釈や文字定数の中のFŌRやNEXTにも反応してしまうので，少しぐあいの悪いこともあるが（右のページの行410〜450参照），REM文の中と 2 重引用符で囲まれた部分をスキップするように改造すれば，これを避けることができる（例題 9 参照）．

行300〜370は行番号を分離するためのプログラムで，「数字以外の文字が来るまでは行番号」という方式で処理しているが，別案としては「最初の空白をさがし，これより左を行番号とみなす」という方法も考えられる．そのようにすればプログラムは少し簡単になる．

```
 10  REM ▒▒▒▒▒▒▒ TOOL2 ▒▒▒▒▒▒▒
 20  REM ▒                  ▒
 30  REM ▒  pretty  printer  ▒
 40  REM ▒                  ▒
 50  REM ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒
 60  CLEAR 8000
 70  DIM A$(300)
 80  OPEN "O",#2,"LPTO:"
 90  REM ▒▒▒▒▒▒▒ プログラムの よみこみ ▒▒▒▒▒▒▒
100  INPUT "ファイルめいは?";F$
110  OPEN "I",#1,"1:"+F$
120  FOR L=1 TO 400
130    LINE INPUT #1,A$(L)
140    PRINT A$(L)
150    ON ERROR GOTO 200
160  NEXT L
170  PRINT "おわり"
180  CLOSE #1
190  END
200  CLOSE #1
230  REM -------- end of file --------
240  N=L-1
250  REM ▒▒▒▒▒▒▒ indentation ▒▒▒▒▒▒▒
260  LEVEL=0
270  REM --- ぎょうばんごう を ぶんり ---
280  FOR L=1 TO N
290    M=LEN(A$(L))
300    IF M=0 GOTO 470
310    FOR I=1 TO M
320      C$=MID$(A$(L),I,1)
330      LC=ASC(C$)
340      IF (LC<&H30)OR(&H39<LC) GOTO 380
350    NEXT I
360    PRINT "おかしい":PRINT A$(L)
370    END
380    MADE=I-1
390    LN=VAL(LEFT$(A$(L),MADE))
400    B$=RIGHT$(A$(L),M-MADE)
410  J=INSTR(B$,"NEXT")
420  IF J>0 THEN LEVEL=LEVEL-1
430  PRINT #2,USING"#####";LN;
440  PRINT #2,SPC(2*LEVEL+1);B$
450  J=INSTR(B$,"FOR")
460    IF J>0 THEN LEVEL=LEVEL+1
470  NEXT L
480  END
```

例題 9　変数名一覧表

ASCII セーブされたプログラムをデーターとして読み，そこに使われている変数名の一覧表を印刷するプログラムを作れ．変数名は ABC 順に並べ，各変数名がどの行で使われているか，ということも調べて印刷せよ．

[解答]　これは簡単そうであるが意外に手間のかかるプログラムである．長いので少しずつ説明する．

```
10 REM ▒▒▒▒▒▒▒▒▒ TOOL3/HS ▒▒▒▒▒▒▒▒▒
20 REM ▒                              ▒
30 REM ▒     へんすうめい いちらんひょう     ▒
40 REM ▒                              ▒
50 REM ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒
60 TIME$="00:00:00"
70 CLEAR 8000
80 DIM A$(300),VN$(100),CR$(100),V(100)
90 OPEN "O",#2,"LPT0:"
100 REM ▒▒▒▒▒▒▒▒▒ ﾌﾟﾛｸﾞﾗﾑの よみこみ ▒▒▒▒▒▒▒▒▒
110 INPUT "ﾌｧｲﾙめいは?";F$
120 OPEN "I",#1,F$
130 FOR K=1 TO 400
140 LINE INPUT #1,A$(K)
150 PRINT A$(K)
160 ON ERROR GOTO 210
170 NEXT K
180 PRINT "おわり"
190 CLOSE #1
200 END
210 CLOSE #1
```

ここまでは，前問（例題７，８）と同じである．プログラムの終りまで読むと，ERRŌRということになって，行210に進む（実際にはエラーでなくて，これが正常のパスである）．

配列**VN$**は変数名（variable name）の格納場所，**CR$**は参照場所（cross reference）の格納場所（格納方法は本書独自の連接法による．詳細後述），**V**は予約語を分離するための目印を記憶する場所である．

```
250 N=K-1
260 REM ---- き゛ょうは゛んこ゛う を ふ゛んり ----
270 VN$(1)="REM"
280 JMAX=1
290 FOR L=1 TO N
300 PRINT A$(L)
310 M=LEN(A$(L))
320 IF M=0 GOTO 710
330 FOR I=1 TO M
340 C$=MID$(A$(L),I,1)
350 LC=ASC(C$)
360 IF (LC<&H30)OR(&H39<LC) GOTO 400
370 NEXT I
380 PRINT "おかしい":PRINT A$(L)
390 END
```

行290の**FŌR**文は，プログラムの第1行から第N行(最後の行)までについてのくりかえしを制御するもので，対応する**NEXT**文は，行710にある．

行300で，これから処理する文の長さ（文字数）を変数**M**に入れている．これは今後たびたび用いられる．

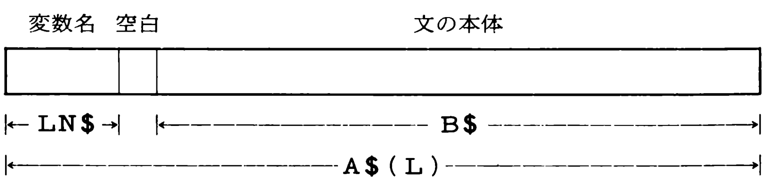
行330～390は行番号を分離するプログラムで，文の左から1字ずつ調べていって，数字以外の文字が出てきたら行400に進む．もしも文の最後まで全部が数字ばかりだとしたら，それはおかしいので，行380でエラー・メッセージを出して停止する．

```
400 MADE=I-1
410 KARA=I+1
420 LN$=LEFT$(A$(L),MADE)
430 LN=VAL(LN$)
440 B$=RIGHT$(A$(L),M-MADE)
```

行400に来たとき，ポインターIは，行番号と文の本体の間の空白の位置にある．ここで文を二つに分け，

第1字～第$I-1$字目（変数名）を**LN$**に入れる．

第$I+1$字目以降　　（文本体）を**B$**　に入れる．



```
450 W$=""
460 SF=0:SQ=1
470 FOR I=KARA TO M
480 IF I<8 AND C$="'" GOTO 700
490 C$=MID$(A$(L),I,1)
500 IF C$=CHR$(&H22) THEN SWAP SF,SQ
510 LC=ASC(C$)
520 IF (&H41<=LC)AND(LC<=&H5A) GOTO 690
530 IF (&H30<=LC)AND(LC<=&H39) GOTO 690
540 IF LEN(W$)=0 GOTO 670
550 IF ASC(W$)<&H41 GOTO 670
560 IF W$="REM" GOTO 710 : REMのよみとばし
570 IF SF=1 GOTO 670
580 FOR J=1 TO JMAX
590 IF W$=VN$(J) GOTO 660
600 NEXT J
610 JMAX=JMAX+1
620 VN$(JMAX)=W$
630 CR$(JMAX)=LN$+" "
640 W$=""
650 GOTO 700
660 CR$(J)=CR$(J)+LN$+" "
670 W$=""
680 GOTO 700
690 W$=W$+C$
700 NEXT I
710 NEXT L
```

変数**W$**は，変数名を組み立てるための作業場所で，最初は空にしておいて，英数字が続いている間は（そのためのチェックが行520〜530である）**W$**に文字をつないでゆく（行690）．英数字以外の文字が来ると行540に進み，先頭が英字であるかどうかを調べる（行550）などの手続きを経て，行580に進む．行580〜600は，その変数名が既に登録されているか（変数名一覧表に入っているか）否かのチェックである．本格的な実用プログラムを作るのであれば，ここでハッシングとか2分法といった技法を使うのであるが，このプログラム例では簡単に，変数名一覧表を上からサーチしている．未登録ということがわかったら変数名一覧表**VN$**に登録する．また（登録済みでも新規でも）参照箇所一覧表**CR$**に，当該行の行番号を連接（文字列の後につなぐこと）する．BASICの文字列変数は可変長なので，こういう場合，いつでも必要なだけの場所を確保できて便利である．

以上が処理の主要部であるが，それだけだと

　　注釈（REM文の中味）が変数名として扱われてしまう

　　文字定数（2重引用符で囲まれた部分）についても同様

という難点があるので，そこだけは読みとばす必要がある．そのための細工が，行460（2重引用符の外にあるか内にあるかを示すフラッグとしてSP,SQを用いている），行480（行頭の ' 印はREMの省略形だから），行500（2重引用符のASCIIコードは22番），行560，行570である．

```
720 GOSUB 930
730 GOSUB 1290
740 FOR J=RW+1 TO JMAX
750 IF V(J)=0 GOTO 770
760 PRINT VN$(J);": ";CR$(J)
770 NEXT J
780 INPUT "ﾌﾟﾘﾝﾄしましょうか?",I$
790 IF I$="N" GOTO 910
800 PRINT #2,"░░░░░░░░░░░";F$;"░░░░░░░░░░░"
810 PRINT #2,"TIME=";TIME$
820 FOR L=1 TO N
830 PRINT #2,A$(L)
840 NEXT L
850 PRINT #2
860 PRINT #2,"░░░░ へんすうめい の いちらんひょう ░░░░"
870 FOR J=RW+1 TO JMAX
880 IF V(J)=0 GOTO 900
890 PRINT #2,VN$(J);": ";CR$(J)
900 NEXT J
910 CLOSE #2
920 END
```

行930〜はABC順に整列するためのプログラム，行1290〜は予約語（IF,FORなどといった綴り）に目印を付ける（Vの値を1にする）プログラムである．これらの処理を行なったあと，ディスプレイ，指示があればプリンターに，ソースリストと結果（変数名一覧表）を出力する．

```
930 REM ********* *********
940 REM ***** SORTING *****
950 REM *** (heap sort) ***
960 REM *******************
```

```
970 FOR L=2 TO JMAX
980 J=L
990 I=INT(J/2)
1000 IF VN$(I)>=VN$(J) GOTO 1040
1010 SWAP VN$(I),VN$(J) : SWAP CR$(I),CR$(J)
1020 J=I
1030 IF J>1 THEN 990
1040 NEXT L
1050 FOR MADE=JMAX TO 2 STEP -1
1060 TAIHI$=VN$(1) : NIGEL$=CR$(1)
1070 I=1
1080 J1=2*I
1090 J2=2*I+1
1100 IF J1=MADE GOTO 1150
1110 IF J1>MADE GOTO 1190
1120 IF VN$(J1)>VN$(J2) GOTO 1150
1130 IO=J2
1140 GOTO 1160
1150 IO=J1
1160 VN$(I)=VN$(IO) : CR$(I)=CR$(IO)
1170 I=IO
1180 GOTO 1080
1190 VN$(IO)=VN$(MADE) : CR$(IO)=CR$(MADE)
1200 I=IO
1210 K=INT(I/2)
1220 IF VN$(K)>=VN$(I) GOTO 1260
1230 SWAP VN$(I),VN$(K) : SWAP CR$(I),CR$(K)
1240 I=K
1250 IF I>1 THEN 1210
1260 VN$(MADE)=TAIHI$ : CR$(MADE)=NIGEL$
1270 NEXT MADE
1280 RETURN
1290 REM ------- reserved words -------
1300 READ RW$
1310 FOR J=1 TO JMAX
1320 IF VN$(J)<RW$ THEN V(J)=1 : GOTO 1400
1330 IF VN$(J)=RW$ GOTO 1370
1340 READ RW$
1350 IF RW$="*" THEN KARA=J : GOTO 1420
1360 GOTO 1320
1370 V(J)=0
1380 READ RW$
1390 IF RW$="*" THEN KARA=J+1 : GOTO 1420
1400 NEXT J
1410 RETURN
1420 FOR J=KARA TO JMAX
1430 V(J)=1
1440 NEXT J
1450 RETURN
```

8 図形表示

8.1 機能紹介

図形を表示するには，最低限，次のことができればよい．

画面を消す．

指定された位置に点（ドット）を表示する．

これさえできれば，あとは工夫しだいでどんな図形でも描けるはずであるが，

指定された2点を直線で結ぶ

指定された点を中心とし，指定された半径で円を描く

指定された位置に，指定された文字を書く

などの機能があれば，さらに便利であろう．また（カラー・グラフィック・ディスプレイやプリンターを持っていれば）

色の指定

プリンターへの出力

などもできるようにする必要がある．

現在市販されているマイコンの大部分の機種は，上記の機能（の全部または一部）を持っており，BASICの命令の形で書くことができる．書き方の詳しい規則は機種によって少しずつ異なるが，基本をマスターしてあれば個々の機種への対応は比較的簡単にできるので，本章では，代表的な二，三の機種を例にとって図形表示関係の文法を説明する．実際にプログラムに書くときの命令の順序は上記のとおりでなく，機種によっては最初に図形表示のための若干の手続きを必要とすることがあるが，それは8.9節でまとめて説明することにして，まずは上記の順に基本事項を説明する．

8.2 画面消去

> これは，普通，
>
> 　　　CLS
>
> と書けばよい（CLSはclear screenの意味である）．この文を実行すると，画面に表示されている文字や図形はすべて消去され，カーソルは画面の左上隅（ホーム・ポジション）に移る．

［**例外**］ 下記の機種にはCLS文がないので，それぞれ，次のように書く．

MZ-80B/C/K の場合* 　PRINT"[CLR]"

（キーボードの[CLR]キーのこと．
上段なのでシフトキーを同時に押す．）

PC-8001 の場合 　PRINT CHR$(12)

M-23 の場合 　PRINT CLEAR

なお，ASCII コードの12番（16進法の 0C 番）は画面消去を表すので，

```
PRINT CHR$(12)
```

または

```
PRINT CHR$(&HC)
```

と書けば，どの機種でも(関数CHR$を使えるならば)共通に，画面消去を行なうことができる．

> **点の消去**　画面全体を消さずに特定の点だけを消したい場合は
>
> 　　　PRESET（横座標，縦座標）
>
> （MZ-80B/C/K の場合は　RESET　横座標，縦座標）を用いる．座標の指定方法は次節で説明する．なお，PRESETのできない機種では背景と同じ色をPSETすることで代用する．

---

* これは SP-5030 などの場合の話．HuBASIC ならばCLSが使える．

8.3 点 の 表 示

> 画面上の指定位置に点を表示するには,普通,次のように書けばよい。
>
> PSET （横座標，縦座標）

［例外］ **Level-3 の場合**　PSET （横座標，縦座標，色番号）

MZ-80B/C/K の場合　SET 横座標，縦座標

M-23 の場合　DISPLAY

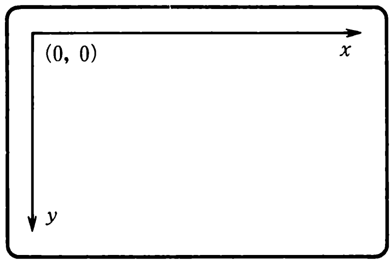
画面座標系　これは普通,

左上隅が原点（0， 0）

横座標は右向きが正

縦座標は下向きが正

で右端，下端の座標は機種によって異なるが，主要機種については下記のようになっている。



機　種	横座標最大値	縦座標最大値	備　考
MZ-80C/K	79	49	
MZ-80B	319	199	V-RAM 使用時
PC-6001	255	191	
PC-8001	159	99	
Basic Master J_R.	255	191	
Level-3	639	199	
Micro-8	639	199	
M-23	639	199	
PASOPIA	639	199	
if-800(モデル20)	639	199	

［例外］ 以上はいずれも絶対座標系であり，座標変換（実際の寸法を上記の画面におさめるための縮尺など）はユーザーが行なうのであるが，PC-8801 のように座標変換を自動的に行なってくれる機種もある。

8.4 直線を引く

直線を引くには，その両端点の座標を指定して，普通，

LINE （横座標，縦座標）－（横座標，縦座標），PSET

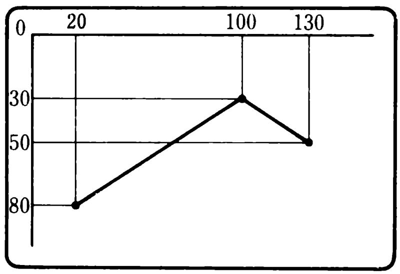
と書く*．第1の（　）内は始点，第2の（　）は終点の座標である．なお，前の線分の終点が次の線分の始点となる場合は始点を省略して

LINE －（横座標，縦座標），PSET

と書いてよい．

(例) LINE （20,80）－（100,30），PSET

LINE －（130,50），PSET



［備考］ LINE文の末尾には，後述のように，色の指定その他各種のオプションを書くことができる．それらのオプションは「書いても書かなくてもよい」という規則になっているのが普通であるが，Level-3 の場合は色の指定を省略できないので注意されたい．

* 中央の－印は，意味としてはハイフンであるが，キーボードにはハイフンに当たる字がないので，負号（マイナス）を用いる．

[**LINE文を使用できない機種の場合**]　ＬＩＮＥ文を使用できない場合は，端点の座標を補間して直線上の点の座標を求め，8.3節で述べた方法で表示してやればよい．そのようなプログラムの一例として，MZ-80C/K のために HuBASIC で書いたサブルーチンを以下に示す．

```
8500 LABEL "SEN"
8510 DX=XJ-XI
8520 DY=YJ-YI
8530 IF ABS(DX)<ABS(DY) GOTO "THEN"
8540 REM ...|DX>DY| ﾉﾊﾞｱｲ...
8550 FOR XX=XI TO XJ STEP SGN(DX)
8560 YY=YI+(XX-XI)*DY/DX
8570 PSET (XX,YY)
8580 NEXT XX
8590 RETURN
8600 REM ...|DX<DY| ﾉ ﾊﾞｱｲ
8610 LABEL "THEN"
8620 FOR YY=YI TO YJ STEP SGN(DY)
8630 XX=XI+(YY-YI)*DX/DY
8640 PSET (XX,YY)
8650 NEXT YY
8660 RETURN
```

[**使用法**]　始点の画面座標をＸＩ，ＹＩに入れ，終点の画面座標をＸＪ，ＹＪに入れて，

　　GŌSUB "SEN"

により，このプログラムを呼び出す．

[**解説**]　始点と終点の間の横座標の差ＤＸと縦座標の差ＤＹを比較し

　　$|DX| \geqq |DY|$ ならばＸを独立変数としてＹを補間

　　$|DX| < |DY|$ ならばＹを独立変数としてＸを補間

している．補間の計算式は6．節で説明したとおりである．行8550および8620のＦŌＲ文の増分欄に書いてあるＳＧＮは3.7節で説明した組込み関数の一つで，引数の符号によって＋１または－１になる．点の表示に，MZ-80 の SP-5030 ならＳＥＴと書くわけであるが，HuBASIC では他機と同様にＰＳＥＴである．

8.5 四 角 を 描 く

ＬＩＮＥ文にはいろいろなオプションを付けられる．まず，基本形

　　　ＬＩＮＥ （横座標，縦座標）－（横座標，縦座標），ＰＳＥＴ

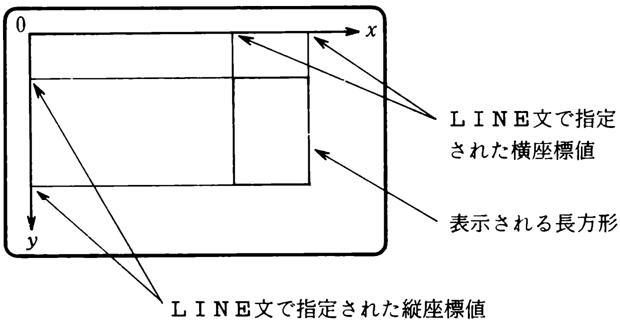
の次に

　　　，色番号

を書くことにより，線の色を指定できる（色番号→8.10節参照）．またその次に

　　　，Ｂ

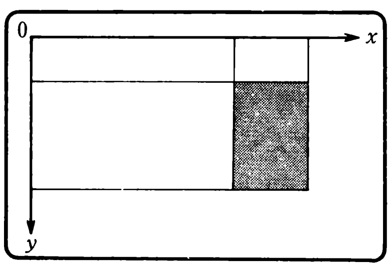
を付けると，普通のＬＩＮＥ文のような直線は引かず，かわりに，



のような長方形が表示される．「，Ｂ」のかわりに

　　　，ＢＦ

とすると，上記の四角の中がぬりつぶされる．



また，ＰＳＥＴのかわりにＰＲＥＳＥＴと書くと，直線または四角が消去される．

8.6 円 を 描 く

（CIRCLE文は使用できない機種がある）

円または円弧を描くにはCIRCLE文を用いれば1命令で書ける．ただし，これを使用できる機種は限られている．使用できる機種は

Micro-8　　*if*-800

VIC-1001

など，書き方は

CIRCLE(中心の横座標，中心の縦座標)，半径，色番号

［CIRCLE文代行プログラム］ CIRCLE文を使用できない機種では，座標を計算してPSETで打点することによりCIRCLE文の代行をするプログラムを作っておくとよい．下記はその一例である．

```
6800 REM ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒
6801 REM ▒       ｻﾌﾞﾙｰﾁﾝ "えん"       ▒
6802 REM ▒ CIRCLE(XO,YO),R,C そうとう ▒
6803 REM ▒       (MB-6890 よう)       ▒
6804 REM ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒
6805 RR=R*R
6806 FOR X1=0 TO 0.707*R
6807 YIS=SQR(RR-X1*X1)
6808 Y1=YIS*0.45
6809 X2=YIS : Y2=0.45*X1
6881 PSET(XO-X1,YO-Y1,C)
6882 PSET(XO-X1,YO+Y1,C)
6883 PSET(XO+X1,YO-Y1,C)
6884 PSET(XO+X1,YO+Y1,C)
6885 PSET(XO-X2,YO-Y2,C)
6886 PSET(XO-X2,YO+Y2,C)
6887 PSET(XO+X2,YO-Y2,C)
6888 PSET(XO+X2,YO+Y2,C)
6889 NEXT X1
6890 RETURN
```

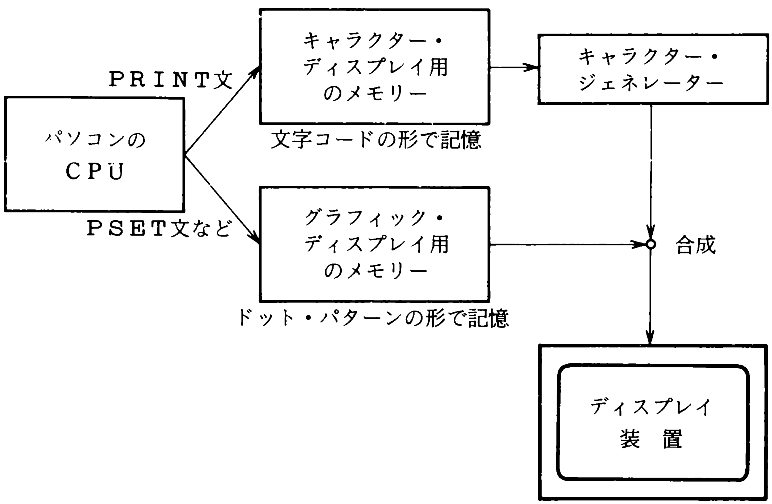
8.7 文字の表示

画面上に文字を表示する方法は二通りある．

- キャラクター・ディスプレイの方式で表示
- グラフィック・ディスプレイの方式で表示

前者はPRINT文による普通の文字出力方式である．それに対し後者は文字をドット・パターンとして（図形の一種として）出力する方式であり，命令としてはPSET文やLINE文を用いてもよいが，機種によっては，更に便利な命令，SYMBŌL文やPUT文を用いることができる．

じつは，パソコンの表示機構は，次の図に示すように，キャラクター・ディスプレイの系統とグラフィック・ディスプレイの系統の二本立てになっている．前記のような区別が生ずるのはそのためである．



プログラムを書く上での大きな差違は，前者の方式（PRINT文による出力）の場合には，画面上の位置の指定に，8.3節で述べた画面座標を用いることができず，

　　上から何行目の，左から何字目

という形で書かなければならない（そのため，二つの座標を併用することになる）ということである．

キャラクター・ディスプレイ方式による表示法　これは

1）表示したい位置*にカーソルを移動させる

2）PRINT文で出力する

という手順で行なう．上記1）はLŌCATE文で行なうのが便利である．

LOCATE文　これは次の形で書く．

LŌCATE　左から何字目，上から何字目

（ただし左端，上端の位置を「0字目」として数える）．これを実行すると，カーソルは指定位置に移動する．

（例）
```
LŌCATE 5,10
PRINT "A"
```

文字座標系　上記の方式で指定したときの画面上の位置は

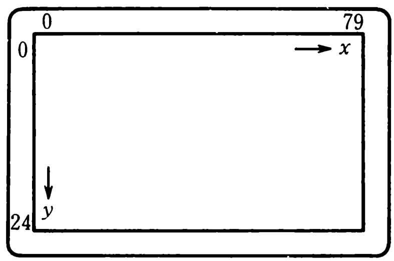
1行に何文字表示するか

1画面に何行表示するか

によって変ってくる．これらは後述のWIDTH文によって何種類か選べるが，標準的なのは

横80字　縦25行

という表示方式で，その場合の



LŌCATE文の指定と画面上の位置

の対応（いわば文字座標系）は上図のようになる．この目盛と，

横方向640ドット　縦方向200ドット

の機種のグラフィック座標系の関係は，横方向，縦方向とも，ちょうど1対8になる（これは文字を8×8ドットで表示しているためである）．したがって，文字座標の値を8倍すれば，その文字の左上隅のドットの画面座標になる．逆に，グラフィック座標を8で割って端数を切り捨てれば文字座標になる．

---

*　正確にいえば「文字列の先頭の文字の表示位置」

グラフィック・ディスプレイ方式による表示法　前ページの方法は，どの機種でも使えて，簡単で，速い，など利点があるが，そのかわり，座標系が二本立てになって不便であり，文字の大きさを変えられない，といった難点もある．それを解決するにはグラフィック・ディスプレイ方式で出力すればよい．

Micro-8 のSYMBŌL文はそのための命令である．これは次のように書く．

> SYMBŌL（横座標，縦座標），文字列式，横倍率，縦倍率
>
> この後に次のようなオプションを付記してもよい．
>
> ，色番号，文字の向き

ここで横座標，縦座標は画面座標（すなわちPSET文やLTNE文と同じ座標系）で，先頭の文字（正確にいえばその文字わく）の左上隅の位置を指定する．文字列式というのは，一般には文字列型の式（たとえばA$+"YEN"）のことであるが，普通は単に文字列型の変数名または文字型定数（すなわち，文字列を2重引用符で囲んだもの）と思ってよい．横倍率と縦倍率は文字の大きさを指定するもので，8×8ドット（1行80字で表示したときの大きさ）を基本単位として，その何倍に拡大するかを書く．拡大文字はタイトルの表示などに使用できる．文字の向きは

	1	2	3	4
	上向き	左向き	下向き	右向き
(例)	A	∢	∀	⊳

で指定する．このような機能は座標軸の目盛りや説明を描くのに便利である．

8.8 GET @ 文と PUT @文

（使用できない機種がある.）

文字や図形を表示する機能の一つとして,

文字や図形のドット・パターンを配列に入れておく

それを画面上の指定位置に表示させる

ということが簡単にできると便利であろう. 用途としては, たとえば

漢字を表示する

回路図の部品記号を表示する

会社や製品のシンボル・マークを表示する

模様を描く

画面をディスクやテープに入れて保存する

などが考えられる. これを行なうのがGET@文とPUT@文である. その使用法には各種のバリエーションがあるので詳しい説明は省略するが, 上記のような目的の場合には次のように書けばよい*.

GET@ （左端，上端），（右端，下端），配列名，G
PUT@ （左端，上端）－（右端，下端），配列名，PSET

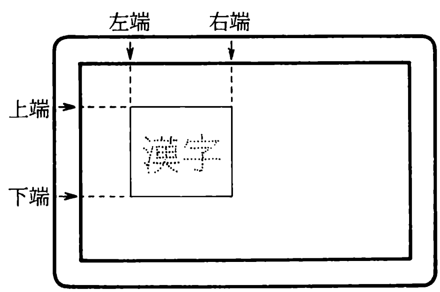
ここで左端, 上端, 右端, 下端は, ドットを読みとる区域, あるいは表示する区域で, 画面座標（横座標と縦座標）で指定する. 配列の型については別に制限はなく,

要するに指定した区域のドット数, すなわち

（右端－左端＋1）

×（下端－上端＋1）

を収容できるだけのビット数を収容できればよい（実際の扱いはバイト単位だから上記の値を8で割り, 端数を切り上げる).



---

* ここの説明はMicro-8用

［**GET @文を代行するプログラム**］ GET＠文やPUT＠文を使用できない機種はかなり多いので，その利用者のために，これらの文の機能を代行するプログラムを作ってみた．

まず，GET＠であるが，GŌSUB7000で下記のサブルーチンを呼び出すと，画面座標の

左端X１　右端X２　上端Y１　下端Y２

内の区域のドット・パターンが，文字列型変数IMAGE$に読み込まれる．格納先を文字列変数としたのは，

文字列変数ならば可変長なので使い易い

代入やディスク入出力などが簡単

必要ならば配列の形で扱うこともできる（224ページの例参照）などの理由による．

```
7000 ' ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒
7010 ' ▒          ｻﾌﾞﾙｰﾁﾝ "GET"          ▒
7020 ' ▒ (X1,Y1)-(X2,Y2)のﾄﾞｯﾄ･ﾊﾟﾀｰﾝ ▒
7030 ' ▒  を もじれつへんすうIMAGE$にかくのうする  ▒
7040 ' ▒            (MB-6890よう)            ▒
7050 ' ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒
7060 GOSUB 7590 : REM constants せってい
7070 IMAGE$=""
7080 P=7
7090 BYTE=0
7100 FOR X=X1 TO X2
7110 FOR Y=Y1 TO Y2
7120 DOT=POINT(X,Y)
7130 IF DOT=0 GOTO 7150
7140 BYTE=BYTE OR BIT(P)
7150 IF P>0 THEN P=P-1 : GOTO 7190
7160 P=7
7170 IMAGE$=IMAGE$+CHR$(BYTE)
7180 BYTE=0
7190 NEXT Y
7200 NEXT X
7210 IMAGE$=IMAGE$+CHR$(BYTE)
7220 RETURN
```

［**解説**］ サブルーチン7590は右のページにあり，１バイトの中の右から*P*番目のビットのみ１，他は０，という定数を作り配列BITに格納している．行7120の関数PŌINT（X，Y）は，画面座標X，Yの点にドットがあれば−１，ドットがなければ０の値をとる組込み関数である．

［**PUT @文を代行するプログラム**］　これは左記のプログラムで文字列変数**IMAGE$**に記憶させたドット・パターンを画面上に再表示するプログラムで，表示区域

左端**X1**　　右端**X2**　　上端**Y1**　　下端**Y2**

および色番号**C**を指定することができる．行7380から7430までは，文字列変数の先頭から１バイトずつとり出し，それをドット・パターンとして扱うための細工，また行7460はその右から P 番目のビットだけをとり出すための細工であり，機械語で処理すれば簡単であるが BASIC でも一応こんなぐあいにして書けるわけである．

```
7300 ' ※※※※※※※※※※※※※※※※※※※※※※※※※※※※※※※※
7310 ' ※          ｻﾌﾞﾙｰﾁﾝ "PUT"              ※
7320 ' ※       IMAGE$のﾋﾞｯﾄ･ﾊﾟﾀｰﾝを           ※
7330 ' ※  (X1,Y1)-(X2,Y2)にﾃﾞｨｽﾌﾟﾚｲ          ※
7340 ' ※            (MB-6890よう)              ※
7350 ' ※※※※※※※※※※※※※※※※※※※※※※※※※※※※※※※※
7360 IF C=0 THEN C=7
7370 GOSUB 7590 : REM constants せってい
7380 MOJIRETU$=IMAGE$
7390 P=7
7400 JISUU=LEN(MOJIRETU$)
7410 IF JISUU=0 THEN RETURN
7420 LM$=LEFT$(MOJIRETU$,1)
7430 BYTE=ASC(LM$)
7440 FOR X=X1 TO X2
7450 FOR Y=Y1 TO Y2
7460 DOT=BYTE AND BIT(P)
7470 IF DOT<>0 THEN PSET(X,Y,C)
7480 IF P>0 THEN P=P-1 : GOTO 7560
7490 P=7
7500 JISUU=JISUU-1
7510 MOJIRETU$=RIGHT$(MOJIRETU$,JISUU)
7520 IF JISUU=0 THEN RETURN
7530 LM$=LEFT$(MOJIRETU$,1)
7540 IF LM$=" " THEN BYTE=0 : GOTO 7560
7550 BYTE=ASC(LM$)
7560 NEXT Y
7570 NEXT X
7580 RETURN
7590 REM --- bit constants ---
7600 IF SETTEIZUMI=1 THEN RETURN
7610 SETTEIZUMI=1
7620 DIM BIT(7)
7630 FOR P=0 TO 7
7640 BIT(P)=2^P
7650 NEXT P
7660 SETTEIZUMI=1
7670 RETURN
```

［**注意**］　行7470の**PSET**文はLevel-3 の書き方になっている．他機種の場合は**PSET（X，Y）**とし，必要に応じてその後に色指定を書けばよい．

8.9 オープニング・セレモニー

機種によっては，図形表示機能（PSET文，LINE文など）を用いる前に，表示モードの設定その他を行なう規則になっているものがある．この種のものは，個々の文の文法的意味を考えるよりも，「図形表示を行なう場合にはプログラムの最初に必ずこの決まり文句を書く」と覚えてしまう方がよいと思う．

［**PC-8001 の場合**］ モノクロ表示の場合は次のように書く．

```
WIDTH 80,25                ←── 1行に80字，画面25行
CŌNSŌLE 21, 5,1,0
                 ↑──────モノクロ・モードの指定
CŌLŌR 0,0,1
          ↑────グラフィック・モードに入ることを示す
```

カラー表示の場合は次のように書く．

```
WIDTH 80,25
CŌNSŌLE 21, 5,1,1
                 ↑──────カラー・モードの指定
CŌLŌR 色番号,0,1
             ↑──────グラフィック・モードを指定
```

色番号は次節で説明する．

［**PC-6001 の場合**］ PC-6001 には 4 種類の表示モードがあるが，1 番は文字モード，2 番は粗すぎてグラフ表示等には向かず，4 番はモノクロになってつまらないので常に 3 番を使うのがよい．その場合の書き方は次のようになる．

```
SCREEN 3,2,2
CŌLŌR 表示色，背景色，コード表の選択
```

CŌLŌR文の指定については次節で説明する．

8.10 色 の 指 定

色番号は普通，次のようになっている．

0	1	2	3	4	5	6	7
黒	青	赤	紫	緑	水色	黄	白

［**例外**］ PC-6001 の表示モード 3（前ページの解説参照）の場合の，番号と色の対応は次のようになっている*．

	1	2	3	4
コード表 1	緑	黄	青	赤
コード表 2	白	水色	紫	橙

コード表 1，2 の内のどちらを使うかは CŌLŌR 文

CŌLŌR 表示色，背景色，コード表の選択

の第 3 のパラメーターで指定する．なお，画面の一部分のみをコード表 1 で表示し他の部分をコード表 2 で指定するというようなことはできない．

［**色と指定の範囲**］ 色は 1 ドットごと独立して指定できることが望ましいが，それには大容量のメモリーを必要とする（たとえば640×200ドットの各点独立に 8 色の指定ができるようにするには 48KB のメモリーが必要**）．そこで，メモリーを節約するため，色の指定を粗くしている（近くの点は同じ色になる）機種が多い．

* この表はマニュアルの記述をそのまま写したものである．実際の色はこれとかなり違い，黄は黄緑に近く，橙は茶に近く，紫はピンクである．

** $2^3=8$ だから 8 色の指定には 3 ビット必要．したがって640×200×3 ビット必要で，これをバイト単位に換算すると48キロバイトになる．

*** 1 ドットごと独立に色指定できる機種は，本書の執筆時点では，Micro-8，PC-8801，*if*-800 など，いわゆる高級機に限られている．

8.11 例題および演習

例題 1 ―――― **線を引く**

ＬＩＮＥ文を用いて，画面に大きく **日本** と表示するプログラムを作れ．

[出題の意図] 図形表示のプログラムを描くにはまずＰＳＥＴ文やＬＩＮＥ文，ＣＩＲＣＬＥ文などを定規やコンパスのように自由自在に使えるように慣れることが大切である．本題にならって，自分の名前，学校名，地名を表示したり，模様を書いたり，マンガを描いたりしてみるとよい．

[解法手順] 大きなグラフ用紙に，画面の１ドット（または２ドット）が１目盛に対応するよう計ってワクと座標を書き，紙の上でデザインを決めて座標を読み取りＬＩＮＥ文で書けばよい．

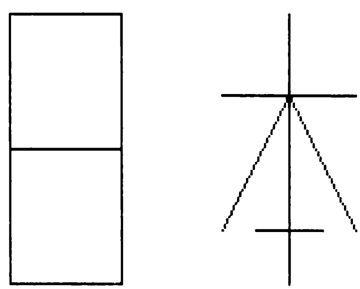
[プログラム例]

```
10  CLS
20  LINE(170, 50)-(170,150),PSET
30  LINE(170, 50)-(270, 50),PSET
40  LINE(270, 50)-(270,150),PSET
50  LINE(170,100)-(270,100),PSET
60  LINE(170,150)-(270,150),PSET
70  LINE(360, 80)-(480, 80),PSET
80  LINE(420, 50)-(420,150),PSET
90  LINE(420, 80)-(360,130),PSET
100 LINE(420, 80)-(480,130),PSET
110 LINE(390,130)-(450,130),PSET
```

[別解]

```
0  CLS
10 A=170 : B=270
20 C= 50 : D=100 : E=150
30 F=360 : G=390 : H=420
40 I=450 : J=480
50 K= 80 : L=130
60 LINE(A,C)-(A,E),PSET
70 LINE-(B,E),PSET
80 LINE-(B,C),PSET
90 LINE-(A,C),PSET
100 LINE(A,D)-(B,D),PSET
110 LINE(F,K)-(J,K),PSET
120 LINE(H,C)-(H,E),PSET
130 LINE(F,L)-(H,K),PSET
140 LINE-(J,L),PSET
150 LINE(G,L)-(I,L),PSET
```

[出力例]



例題 2 ―――― スケーリング

一組のデーター

$$\begin{array}{ccccccc} x_1 & x_2 & x_3 & \cdots & x_i & \cdots & x_n \\ y_1 & y_2 & y_3 & & y_i & & y_n \end{array}$$

が与えられたとき，これを散布図の形式で画面上に表示するプログラムを作れ．

［**解答**］ 散布図というのは各 i について，座標 (x_i, y_i) の点をプロットしたものであるから，画面におさまるようにスケーリングして，

```
FOR I=1 TO N
    PSET(X(I),Y(I))
NEXT I
```

という要領で書けばよい．

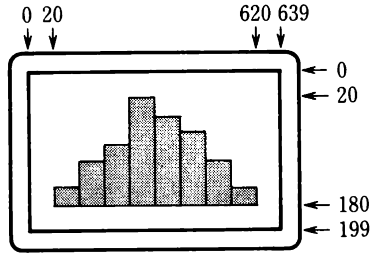
［**プログラム例**］

```
10 INPUT"N=";N
20 DIM X(N),Y(N)
30 FOR I=1 TO N
40 PRINT I;TAB(5);
50 INPUT X(I),Y(I)
60 NEXT I
70 XMIN=X(1) : XMAX=X(1)
80 YMIN=Y(1) : YMAX=Y(1)
90 SX=X(1) : SY=Y(1)
100 FOR I=2 TO N
110 IF X(I)<XMIN THEN XMIN=X(I) ELSE IF X(I)>XMAX THEN XMAX=X(I)
120 IF Y(I)<YMIN THEN YMIN=Y(I) ELSE IF Y(I)>YMAX THEN YMAX=Y(I)
130 SX=SX+X(I) : SY=SY+Y(I)
140 NEXT I
150 CX=0.99*640/(XMAX-XMIN)
160 CY=0.99*200/(YMAX-YMIN)
170 XM=SX/N : YM=SY/N
180 CLS
190 FOR I=1 TO N
200 XX=640-(X(I)-XMIN)*CX
210 YY=200-(Y(I)-YMIN)*CY
220 PSET (XX,YY,6)
230 NEXT I
240 XX=640-(XM-XMIN)*CX
250 YY=200-(YM-YMIN)*CY
260 LINE(0,YY)-(639,YY),PSET,2
270 LINE(XX,0)-(XX,199),PSET,2
```

例題 3 ―― 棒グラフ

データー y_1, y_2, …, y_n を読み込み，棒グラフで表示するプログラムを作れ．ただし y は負の値をとらないものとする．

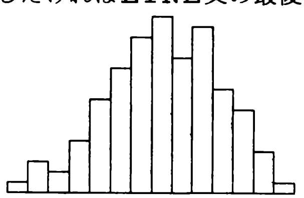
［**解答**］ このような問題の場合，まず方眼紙の上で画面のデザインをするとよい．640×200ドットの画面であれば，棒グラフの横幅を600ドットにとり，基準線を y ＝180とし，縦方向が160ドットにおさまるようにスケーリングするとよいと思う．こうして座標変換の式を決めればあとは簡単で，8.5節の方法を応用して描けばよい．



［**プログラム例**］

```
10 INPUT"N=";N
20 DIM Y(N)
30 YMAX=0
40 FOR I=1 TO N
50 PRINT I,
60 INPUT Y(I)
70 IF Y(I)>YMAX THEN YMAX=Y(I)
80 NEXT I
90 COLOR 1,7
100 CLS
110 C=180/YMAX
120 H=600/N
130 FOR I=1 TO N
140 XR=H*I
150 XL=XR-H
160 YY=180-Y(I)*C
170 LINE(XL,180)-(XR,YY),PSET,1,B
180 NEXT I
```

［**出力例**］ （ぬりつぶしたければLINE文の最後をBFにすればよい）



例題 4 ―――― 折れ線グラフ

データー y_1, y_2, …, y_n を読み込み，折れ線グラフ(x 方向は等間隔とする)で表示するプログラムを作れ．ただし y の値は負の値をとらないものとする．

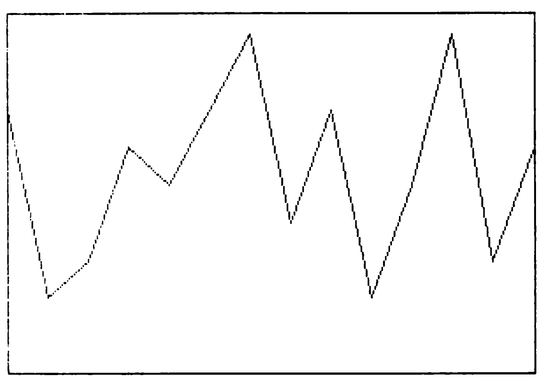
[**解答**] 前問と同様に，画面座標と x, y の値の関係をよく考えておけば，あとは簡単である．

[**プログラム例**]

```
10 INPUT"N=";N
20 DIM Y(N)
30 YMAX=0
40 FOR I=1 TO N
50 PRINT I,
60 INPUT Y(I)
70 IF Y(I)>YMAX THEN YMAX=Y(I)
80 NEXT I
90 COLOR 1,7
100 CLS
110 C=160/YMAX
120 H=600/(N-1)
130 FOR I=1 TO N-1
140 XR=H*I+20
150 XL=XR-H
160 YR=190-Y(I+1)*C
170 YL=190-Y(I)*C
180 LINE(XL,YL)-(XR,YR),PSET,2
190 NEXT I
200 LINE(20,20)-(620,190),PSET,1,B
```

最後の**LINE**文は，ワクを表示するためである．

[**出力例**]



例題　5　――― $y=f(x)$ のグラフ

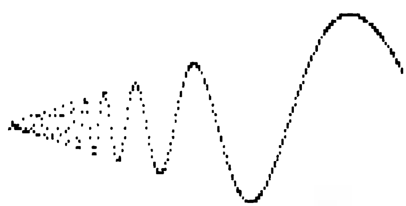
0.001≦x≦0.3の範囲で $y=\dfrac{\sin x}{x}$ のグラフを表示するプログラムを作れ。

［解答］　横方向を1ドット分ずつ動かし，それに対応する y の値を計算しスケーリングして表示すればよいが，PSET文による打点だけだと変動の激しい部分で点がとびとびになってしまう．そのような部分では隣の点との間をLINE文で結ぶとよい．以下に示すのはそういうプログラムの一例である．

```
10 DEF FNC(X)=SIN(1/X)*X
20 XMIN=0.001
30 XMAX=0.3
40 YMIN=-0.3
50 YMAX=+0.3
60 REM -------- ここまで゛ はユーザ゛ ーが゛ かく --------
70 COLOR 7,0
80 CLS
90 C=160/(YMAX-YMIN)
100 DX=(XMAX-XMIN)/600
110 FOR I=1 TO 599
120 X=DX*I+XMIN
130 XX=I+20
140 Y=FNC(X)
150 YY=185-(Y-YMIN)*C
160 IF I=1 GOTO 200
170 IF ABS(YY-YQ)<2 GOTO 200
180 LINE(XQ,YQ)-(XX,YY),PSET,6
190 GOTO 210
200 PSET(XX,YY,6)
210 XQ=XX : YQ=YY
220 NEXT I
230 LINE(20,20)-(620,190),PSET,4,B
```



上記のプログラムによる出力例



PSET 文だけで描いた場合

例題 6 ―― 極座標

極座標で表された曲線

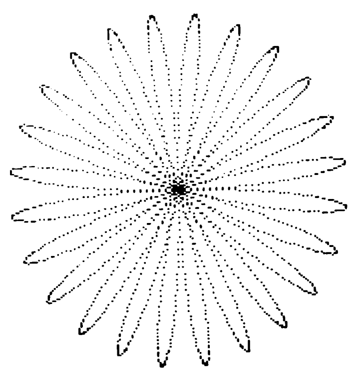
$$r = \sin 12\theta \qquad (0 \leqq \theta \leqq 2\pi)$$

を描くプログラムを作れ．

[**解答**] このような図形を描く場合には，横方向の係数CXと縦方向の係数CYの比をうまく調節する必要がある．それにはマニュアルを研究して決めてもよいが，ひとまず1:1で表示し，その寸法を計って係数を決めるのが簡単である．

```
10 DEF FNR(T)=SIN(12*T)
20 A=0
30 B=2*3.1416
40 N=2000
50 REM --- ここまでﾞはユーザﾞーがﾞかく ---
60 DIM R(N)
70 DT=(B-A)/N
80 RMAX=0
90 REM かんすうちのけいさん と スケーリンクﾞのけいすうのさんてい
100 FOR I=0 TO N
110 T=I*DT
120 R(I)=FNR(T)
130 AR=ABS(R(I))
140 IF AR>RMAX THEN RMAX=AR
150 LOCATE 0,0 : PRINT I;
160 NEXT I
170 CY=80/RMAX
180 CX=CY*2.5
190 REM --- ﾃﾞｨｽﾌﾟﾚｲ ---
200 CLS
210 FOR I=0 TO N
220 T=I*DT
230 XX=320+CX*R(I)*COS(T)
240 YY=110-CY*R(I)*SIN(T)
250 PSET (XX,YY,6)
260 NEXT I
```

[**出力例**]



例題 7 —— 円グラフ

百分比 a_1, a_2, …, a_n を入力し，円グラフで表示するプログラムを作れ.

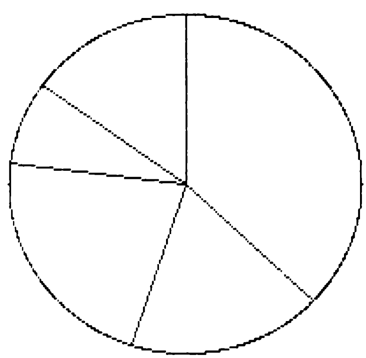
［**解答**］ 円グラフを描くには，まず円を描き，次に

$$s_1 = a_1 \quad s_2 = a_1 + a_2 \quad s_3 = a_1 + a_2 + a_3 \quad \cdots \quad s_n = a_1 + \cdots + a_n$$

を計算し，それに対応する角度を計算し，**LINE**文で線を引き，1点ごとに色指定のできる機種であれば**PAINT**文で色付けすればよい.

```
10 INPUT"N=";N
20 DIM A(N),S(N)
30 FOR I=1 TO N
40 PRINT I,
50 INPUT A(I)
60 NEXT I
70 COLOR 5,1:C=7:CLS
80 F=6.28/100
90 X0=320
100 Y0=100
110 R=80*2.5
120 GOSUB 230
130 LINE(320,100)-(320,20),PSET
140 S(1)=A(1)
150 FOR I=2 TO N
160 S(I)=S(I-1)+A(I)
170 T=F*S(I)
180 X=320+R*SIN(T)
190 Y=100-R*COS(T)/2.5
200 LINE(320,100)-(X,Y),PSET
210 NEXT I
220 END
230 RR=R*R
240 FOR X1=0 TO 0.707*R
250 YIS=SQR(RR-X1*X1)
260 Y1=YIS/2.5
270 X2=YIS : Y2=X1/2.5
280 PSET(X0-X1,Y0-Y1,C)
290 PSET(X0-X1,Y0+Y1,C)
300 PSET(X0+X1,Y0-Y1,C)
310 PSET(X0+X1,Y0+Y1,C)
320 PSET(X0-X2,Y0-Y2,C)
330 PSET(X0-X2,Y0+Y2,C)
340 PSET(X0+X2,Y0-Y2,C)
350 PSET(X0+X2,Y0+Y2,C)
360 NEXT X1
370 RETURN
```

［**出力例**］



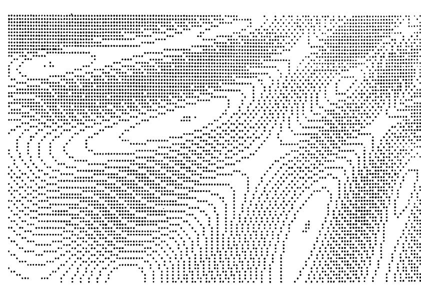
例題 8 ―― 等高線

2 変数関数 $f(x, y)$ の等高線（等値線）を描くプログラムを作れ．

［解答］ かなり細かい格子点上で関数値を計算し，隣り合う 2 点の関数値の間の等高線として表示すべき値が含まれていればプロットする．

```
10 DEF FNC(X,Y)=SIN(X-Y*Y)*SIN(Y-X*X)+1/(0.5+X*X+Y*Y)
20 XMIN=-1 : XMAX=2.5
30 YMIN=0 : YMAX=2.5
40 FMIN=-1 : FMAX=1
50 REM --- ここまて゛は ユーサ゛ーか゛かく ---
60 CLS
70 DX=(XMAX-XMIN)/600
80 DY=(YMAX-YMIN)/160
90 W=(FMAX-FMIN)/20
100 FOR J=20 TO 180 STEP 2
110 Y=YMIN+(180-J)*DY
120 FOR I=20 TO 620 STEP 4
130 X=XMIN+(I-20)*DX
140 Z=FNC(X,Y)
150 T=INT((Z-FMIN)/W)
160 IF T<>TQ THEN PSET(I-2,J,6)
170 TQ=T
180 NEXT I
190 PRESET(18,J)
200 NEXT J
210 FOR I=22 TO 618 STEP 4
220 X=XMIN+(I-20)*DX
230 FOR J=21 TO 179 STEP 2
240 Y=YMIN+(180-J)*DY
250 Z=FNC(X,Y)
260 T=INT((Z-FMIN)/W)
270 IF T<>TQ THEN PSET(I,J-1,6)
280 TQ=T
290 NEXT J
300 PRESET(I,20)
310 NEXT I
```

［出力例］



例題 9 ―― 動かす

（3針表示方式の）時計のプログラムを作れ．

［**解答**］ 高速のアニメーションは機械語で書かないと無理であるが，時計ぐらいならBASICでも書ける．以下に示すのはPC-6001のプログラムの例である．

まず行10で現在時刻（*H* 時 *M* 分）を入力する．行20～40はオープニング・セレモニーで，画面は水色になる．

行50および行210～260は内蔵タイマーの値を組込み関数TIMEで読んで現在時刻（*H* 時 *M* 分 *S* 秒）のデーターを更新するプログラムである．PC-6001の内蔵タイマーは1/1024秒単位の整数表示という，他機種となかなり違った方式をとっており，そのため，現在時刻の算定に手間がかかっている（ここの部分はNEC編「みんなで使おうBASIC」のプログラム例を参考にした）．

文字盤の中心は画面座標（100，100）とし，長針と短針の先端の座標は（XM，YM）と（XH，YH）で表すことにした．LINE文により長針と短針の表示を行なっているのは行700～780のプログラムで，

行700　今まで表示されていた長針を消す

行710～730　長針の新位置を計算して表示

行740　今まで表示されていた短針を消す

行750～770　短針の新位置を計算して表示

という手順によって針を進めている．最初は「今まで表示されていた針」がないので，おかしなことにならぬよう，行60～80で初期設定を行なっている．行100～140では文字盤の円を書いている（もっと速くしたければ8.6節のように書き替えるとよい）．行200～410が反復部分で，まず，秒の値が変るのを待ち，変ったら，それまでの秒針を消して，秒針の新位置を計算して表示している（行330～400）．なお，行270～320は，画面の一部に時刻をデジタル表示するプログラムである（行270，280は今までに表示されていた文字を消す操作）．

```
0 REM PC-6001の とけい
10 INPUT"いまなんじ?";H,M
20 SCREEN 3,2,2
30 COLOR 1,2,2
40 CLS
50 ST=TIME
60 X=100 : Y=100
70 XM=100 : YM=100
80 XH=100 : YH=100
90 GOSUB 700
100 FOR A=0 TO 6.28 STEP 0.05
110 XX=70*SIN(A)+100
120 YY=70*COS(A)+100
130 PSET(XX,YY),1
140 NEXT A
200 REM くりかえし
210 SQ=S
220 NT=TIME
230 S=INT((NT-ST)/976)
240 IF S=SQ THEN 220
250 IF S>60 THEN M=M+1:S=0:ST=TIME:GOSUB700
260 IF M>=60 THEN H=H+1 : M=0
270 LINE(120,0)-(70,10),2,BF
280 IF S=0 THEN LINE(0,0)-(130,1),2,BF
290 LOCATE 0,0
300 PRINT TAB(1);H;":";
310 PTINT TAB(4);M;":";
320 PRINT TAB(7);S
330 LINE(XX,YY)-(X,Y),2
340 DX=SIN(0.105*S)
350 DY=-COS(0.105*S)
360 XX=100+60*DX
370 YY=100+60*DY
380 X=100+70*DX
390 Y=100+70*DY
400 LINE(XX,YY)-(X,Y),3
410 GOTO 200
700 LINE(100,100)-(XM,YM),2
710 XM=100+60*SIN(0.105*M)
720 YM=100-60*COS(0.105*M)
730 LINE(100,100)-(XM,YM),4
740 LINE(100,100)-(XH,YH),2
750 XH=100+50*SIN(0.523*H)
760 YH=100-50*COS(0.523*H)
770 LINE(100,100)-1XH,YH),4
780 RETURN
```

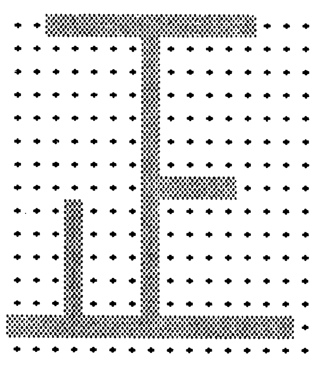
例題 10　漢字のフォント作成

16×16のドット・パターンで漢字を表示するためのフォントを画面上でデザインして，文字列型配列KANJI＄に格納するプログラムを作れ．

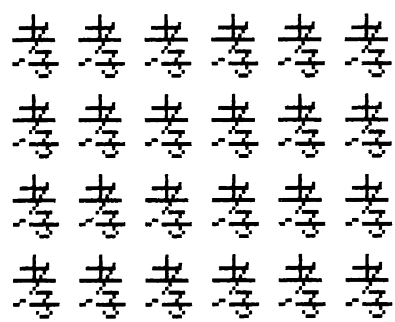
［**解答**］ 簡単なプログラム例を以下に示す．このプログラムを実行すると，画面上に小さな点が16行16列表示される（行6800〜6950）．その内の1点だけが赤く光っている．これがカーソルの代用品で，カーソル移動キー［→］［↑］［←］［↓］を押すと矢印の向きに赤い点が動く．赤い点が希望位置（ここにドットを表示したいという位置）に来たら，Sのキーを押すと，その位置が配列DPに記録され，画面には水色の四角印が表示される（左下の図参照）．既に登録，表示されているドットを取り消したい場合は，そこに赤い点を合わせてRのキーを押す．こうしてフォント（1字分）が完成したらIのキーを押すと，

コート゛を　なんは゛んにしますか

という表示が出るから，その漢字に付けたい番号を入力する（それが配列KANJI＄の添字になる）．このようにして，いろいろな漢字のフォントを作り，漢字 ROM ならぬ漢字 RAM に入れてゆく．全部でき上ったならば，16×16のドットが表示された時点でEのキーを押すと，このプログラムを脱出してメインプログラムに帰る．



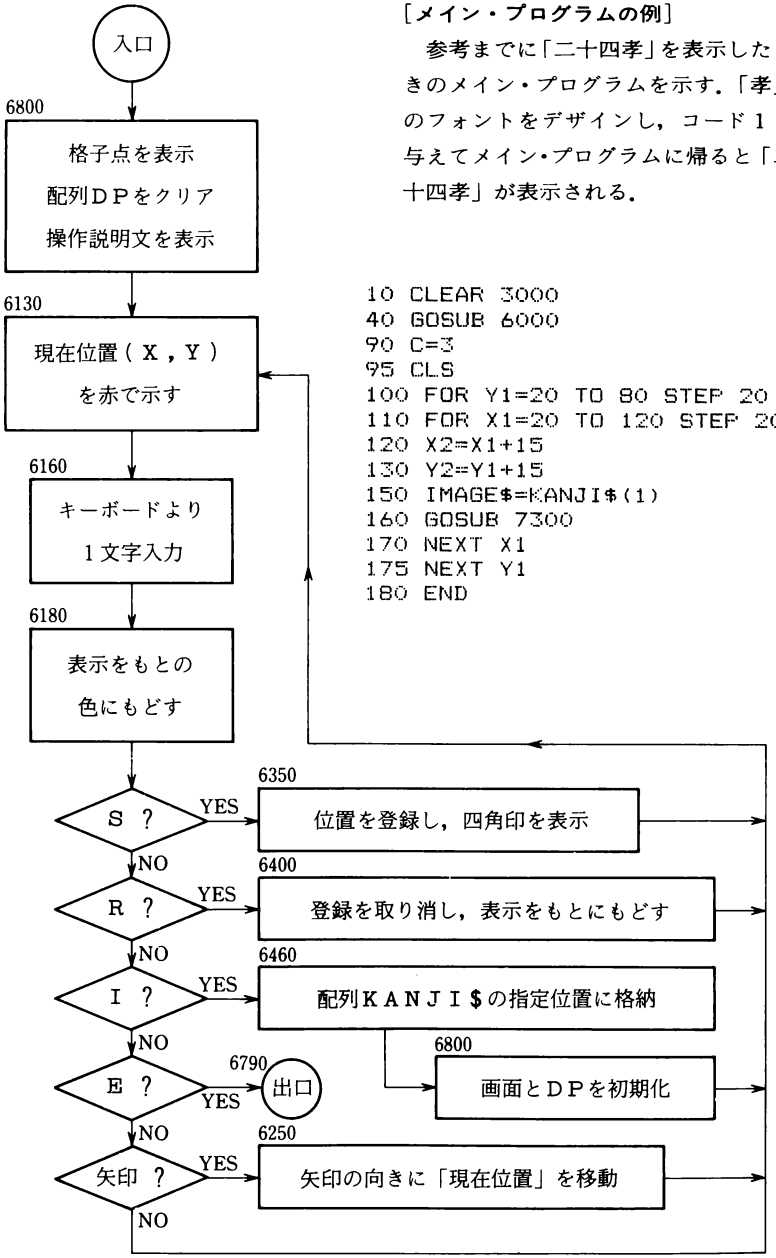
フォント作成中の画面



このプログラムで作成した
フォントによって表示した
「二十四孝」

```
6000 REM ※※※※※※※※※※※※※※※※※※※※※※※※※※
6010 REM ※                        ※
6020 REM ※   ｻﾌﾞﾙｰﾁﾝ "FONTさくせい"   ※
6030 REM ※       (MB-6890 よう)      ※
6040 REM ※※※※※※※※※※※※※※※※※※※※※※※※※※
6050 IF INI=1 GOTO 6100
6060 DIM DP(15,15)
6070 DIM KANJI$(20)
6080 INI=1
6090 ADR=0
6100 CONSOLE 17,8
6110 GOSUB 6790 : REM こうしてんをひょうじ
6120 REM --- ﾊﾟﾀｰﾝよみこみ ---
6130 JI=SCREEN(X,Y)
6140 LOCATE X,Y,1
6150 COLOR 2 : PRINT CHR$(JI);
6160 J$=INPUT$(1)
6170 LOCATE X,Y
6180 IF DP(X,Y)=0 THEN COLOR 7 : PRINT "・";
6190 IF DP(X,Y)=1 THEN COLOR 5 : PRINT "※";
6200 I=ASC(J$)
6210 IF J$="S" THEN GOSUB 6350
6220 IF J$="R" THEN GOSUB 6400
6230 IF J$="I" THEN GOSUB 6460 : GOSUB 6790
6240 IF J$="E" GOTO 6770
6250 IF I=&H1C THEN X=X+1
6260 IF I=&H1D THEN X=X-1
6270 IF I=&H1E THEN Y=Y-1
6280 IF I=&H1F THEN Y=Y+1
6290 IF X<0 THEN X=0
6300 IF Y<0 THEN Y=0
6310 IF X>15 THEN X=15
6320 IF Y>15 THEN Y=15
6330 GOTO 6130
6340 REM --- BIT IN ---
6350 COLOR 5
6360 LOCATE X,Y
6370 PRINT"※"
6380 DP(X,Y)=1
6390 RETURN
6400 REM --- ERASE ---
6410 COLOR 7
6420 LOCATE X,Y
6430 PRINT"・"
6440 DP(X,Y)=0
6450 RETURN
```

```
6460 REM --- メモる ---
6470 GOSUB 7590 : REM constants せってい
6480 LOCATE 0,16,3
6490 PRINT "コート゛を なんは゛んにしますか(ADR=";ADR;")";
6500 INPUT I$
6510 IF I$<>"" THEN ADR=VAL(I$)
6520 P=7
6530 BYTE=0
6540 FOR X=0 TO 15
6550 FOR Y=0 TO 15
6560 LOCATE X,Y,1
6570 PRINT " ";
6580 DOT=DP(X,Y)
6590 IF DOT=0 GOTO 6610
6600 BYTE=BYTE OR BIT(P)
6610 IF P>0 THEN P=P-1 : GOTO 6650
6620 P=7
6630 IMAGE$=IMAGE$+CHR$(BYTE)
6640 BYTE=0
6650 NEXT Y
6660 NEXT X
6670 IMAGE$=IMAGE$+CHR$(BYTE)
6680 KANJI$(ADR)=IMAGE$
6690 ADR=ADR+1
6700 X1=0:X2=X1+15
6710 Y1=0:Y2=Y1+15
6720 C=6
6730 GOSUB 7300
6740 INPUT WAIT 6750;10,"OK?";I$
6750 IMAGE$=""
6760 RETURN
6770 REM --- end of font ---
6780 RETURN
6790 REM --- こうしてん を ひょうし゛ ---
6800 CLS
6810 COLOR 7
6820 FOR X=0 TO 15
6830 FOR Y=0 TO 15
6840 LOCATE X,Y
6850 PRINT "・";
6860 DP(X,Y)=0
6870 NEXT Y
6880 NEXT X
6890 LOCATE 0,0,1
6900 X=0 : Y=0
6910 LOCATE 0,17
6920 PRINT "Sをおすと ト゛ットか゛ セットされます"
6930 PRINT "Rをおすと ト゛ットをけすことか゛ て゛きます"
6940 PRINT "Iをおすと そのもし゛ か゛ とうろくされます"
6950 PRINT "Eをおすと しゅうりょうになります"
6960 RETURN
```



［**メイン・プログラムの例**］

参考までに「二十四孝」を表示したときのメイン・プログラムを示す．「孝」のフォントをデザインし，コード１を与えてメイン・プログラムに帰ると「二十四孝」が表示される．

```
10 CLEAR 3000
40 GOSUB 6000
90 C=3
95 CLS
100 FOR Y1=20 TO 80 STEP 20
110 FOR X1=20 TO 120 STEP 20
120 X2=X1+15
130 Y2=Y1+15
150 IMAGE$=KANJI$(1)
160 GOSUB 7300
170 NEXT X1
175 NEXT Y1
180 END
```

問　題

8.1（ヒストグラム） N 人の生徒の試験の点数（100点満点とする）を読み込み，

0～4点　5～9点　10～14点　15～19点

20～24点　　　　90～94点　95点以上

に分けて人数を数え，点数の分布を棒グラフで出力するプログラムを作れ．

8.2（年齢別人口構成図） 地方自治体の，男女別，年齢別の人口を読み込み，下記のように図化するプログラムを作れ．

```
90歳以上          *I**
80～89           **I***
70～79          ***I***
60～69       ******I******
50～59       ******I*****
40～49      *******I*******

              男 ←―― ――→ 女
```

8.3（管理図） データー a_1, a_2, …, a_n を読み，その平均値 μ と標準偏差 σ を計算し，グラフの左端が $\mu-6\sigma$，右端が $\mu+6\sigma$ になるようにスケーリングしてプロットせよ．

8.4（対数目盛） データー a_1, a_2, …, a_n を読み，対数目盛でプロットするプログラムを作れ，ただし値の範囲は 10^{-5} から1までとする．

8.5（関数のグラフ） 次の関数のグラフを描け．

(i) $y=x^2$　　($x=0$ から $x=1$ まで)

(ii) $y=\sin x$　　($x=0$ から $x=\pi$ ラジアンまで)

(iii) $y=1/(1+x^2)$　　($x=-3$ から $x=+3$ まで)

(iv) $y=\sqrt{1-x^2}$　　($x=-1$ から $x=+1$ まで)

(v) $y=\sin x-0.3\sin(5x+0.2)$　　($x=0$ から $x=2$ まで)

付　録　　周辺装置の使用法

A.1　プリンター

MZ-80　　メーカー提供の BASIC では

```
PRINT/P　出力項目の列
```

(例)　PRINT/P A,B,C

と書く．Hu BASIC の場合には

```
PRINT #1,出力項目の列
```

と書く．

PC-6001, PC-8001, PC-8801 などでは

```
LPRINT　出力項目の列
```

(例)　LPRINT A,B,C

と書く．

Level-3, Micro-8 などでは，あらかじめ

```
ŌPEN "Ō",#番号,"LPT0:"
```

によって，プリンターに番号を対応させ（同時にバッファーを用意し）

```
PRINT #番号,出力項目の列
```

により出力し，最後に

```
CLŌSE #番号
```

によって，番号との対応を解除する（同時にバッファーも解消される）．

(例)

```
ŌPEN "Ō",#1,"LPT0:"
INPUT A,B
C=A+B
PRINT #1,"コタエ";C
CLŌSE #1
```

A.2　カセット・テープ

1．プログラムをセーブ* する方法

SAVE "ファイル名"

というのが普通の書き方である．ファイル名というのは，後日，再入力するときの識別用に付けておく名前で，英字から始まる6字以内の英数字の列を用いておけば安全である（実際には使用できる文字の範囲はかなり広く，カナ文字や特殊記号を許す機種も多く，7字以上を許す機種も少なくないが）．なお，PC-6001とPC-8001では

CSAVE "ファイル名"

とする．

2．プログラムをロード する方法**

LŌAD "ファイル名"

というのが普通の書き方であるが，PC-6001とPC-8001では

CLŌAD "ファイル名"

とする．ファイル名を省略して，単に

LŌAD　　（またはCLŌAD）

とすると，ファイル名の照合をせずに，最初に見つかったプログラムがロードされる（ただしPC-8001は，これを認めていないので常にファイル名が必要である）．

3．データーの入出力

MZ-80　メーカー提供のBASICでは

出力　PRINT/T　出力項目の列

入力　INPUT/T　入力項目の列

と書く．

---

*　save．保存のためにカセット・テープなどに記録すること

**　load．読み込んで実行可能な状態にすること

Hu BASIC/mz, PC-8001

出力 PRINT #−1，出力項目の列

入力 INPUT #−1，入力項目の列

Level-3, Micro-8

あらかじめ

出力 ŌPEN "Ō"，#番号，"CAS0："

入力 ŌPEN "I"，#番号，"CAS0："

によって，カセット・テープに番号を対応させておいて

出力 PRINT #番号，出力項目の列

入力 INPUT #番号，入力項目の列

により入出力を行ない，入力または出力が終ったら

（入出力とも） CLŌSE #番号

によって，番号との対応を解除する（出力の場合，同時に，バッファーに残っている未記録データーがテープに書き込まれ，最後に end of file のコードが書き込まれる．

4．プログラムを文字列データーとして入力する方法*

BASIC の文字列処理機能を用いてプログラムを解析したり自動書きかえを行なったりするには，一旦，カセット・テープにセーブして，それをデーターとして読み込めばよい．ただし，そのためには

SAVE "ファイル名"，A

という形でセーブする必要がある．末尾の A は「ASCII コード（いいかえれば標準的な文字列データーの形式）で記録せよ」という指示を表す．このような形式でセーブしておけば，データーとして読むことができ，INPUT#文で必要な字数だけ読み込んで処理を行なうことが可能になる．

5．文字列データーとして記録したプログラムを実行する方法*

これは，普通の LŌAD 文でロードすればよい．

---

* PC-8001 のカセット・システムではできない．

A.3 フロッピー・ディスク

プログラムのロード，セーブとデーターの入出力(順編成ファイル* の場合）はカセット・テープとだいたい同じ要領である．ただ，少し違う点は，カセット・テープは普通，1台だけであるのに対し，ディスクは複数台使用することが多いということで，それらを識別するためにドライブ番号（装置番号）というものがある．ドライブ番号は固定しており，フロッピー・ディスク挿入口の所に刻印されている．装置番号の指定を省略すると，普通，最も若い（小さい）番号の装置が指定されたことにして処理される．

1．プログラムのセーブとロード

SAVE "ドライブ番号：ファイル名"

LŌAD "ドライブ番号：ファイル名"

2．データーの入出力（順編成ファイル* の場合）

カセット・テープの場合と同様に

あらかじめŌPEN文によりファイルを開いておく

出力はPRINT#文 ｝で行なう
入力はINPUT#文 ｝

終ったらCLŌSE文によりファイルを閉じる

という要領でプログラムを書けばよい．ŌPEN文の書き方は，

MZ-80

出力　WŌPEN　#番号，"$FDドライブ番号：ファイル名"

入力　RŌPEN　#番号，"$FDドライブ番号：ファイル名"

---

* sequencial file．カセット・テープに記録するのと同じように，データーを1列に並べて記録してゆき，入力時には列の先頭から順に読んでゆく方式のファイル．

PC-8001, PC-8801

出力 ŌPEN "ドライブ番号:ファイル名" FŌR

(前行の続き) ŌUTPUT AS #番号

入力 ŌPEN "ドライブ番号:ファイル名" FŌR

(前行の続き) INPUT AS #番号

[**備考**] PC では,「既存のファイルの後に追加」ということができる.

追加出力 ŌPEN "ドライブ番号:ファイル名" FŌR

(前行の続き) APPEND AS #番号

Level-3, Micro-8

出力 ŌPEN "Ō",#番号,"ドライブ番号:ファイル名"

入力 ŌPEN "I",#番号,"ドライブ番号:ファイル名"

である.

PRINT#文とINPUT#文の書き方は,#番号が付くだけで,あとは普通のPRINT文,INPUT文と同じである.

CLŌSE文は,

CLŌSE #番号

と書けばよい.

[**注意**] 1) PC-8001 の場合(および PC-8801 を N-BASIC で使用する場合)は,ディスクを入れたら,

MŌUNT ドライブ番号

というコマンドによって,パソコンに「ディスクを入れましたよ」ということを教えてやらないといけない.また,ディスクをとり出す前には必ず

REMŌVE ドライブ番号

というコマンドによって,パソコンに「ディスクをとり出しますよ」ということを教えてやらないといけない.これを忘れると,記録内容が失なわれたり,システムが動かなくなったりする.

2) Level-3 の場合,CLŌSEを忘れたり,実行しないうちにディスクをとり出したりすると,ディスクの内容が破壊されることがある.CLŌSEやREMŌVEは,やりすぎて悪いことはないので,ディスクをとり出す前に必ず行なう習慣を付ける必要がある.

3．ランダム・アクセスの方法

ランダム・アクセスというのは，データーを勝手な順序で読み書きすることである（これに対し常に一定の順序で読み書きをすることを順次アクセスという）．ランダム・アクセスを行なうためには，記憶場所に番地を付けておいて，指定された番地のデーターをとり出したり，指定された番地に書き込んだりできるようにすればよい．しかし，ディスクの場合，数百キロバイトの記憶場所の1バイトごとに番地を付けるのは大変だし，1バイト単位で入出力を行なうのは待時間の関係で不経済なので，もっと大きな処理単位を設けて，その単位ごとに番地を付け，まとめて入出力を行なっている．この処理単位を，ディスクの構造上の用語では**セクター**と呼び，プログラマーの用語では**レコード**（記録単位，詳しくは物理的記録単位）という．

レコード・サイズ

MZ-80　32バイト
Level-3　128バイト
PC-8001，PC-8801，Micro-8　256バイト

ディスクの入出力の際には常に

出力時　個々のデーターをレコード単位にまとめる
入力時　レコード単位で読み出したデーターを個々のデーターに仕分けする

という作業が必要になる（このための作業場所を**バッファー**という）．順編成方式の場合には，これらの作業はすべてシステムがやってくれる．PRINT#文で出力されたデーターは，ひとまずバッファーに入り，1レコードぶんまとまると自動的にディスクに転送される．入力の場合も同様に自動的に処理されるので，利用者はバッファーを意識する必要がない．しかし，ランダム・アクセスの場合には，自動的にやってくれないので，利用者が自分でプログラムを書く必要がある．

Microsoft 系の BASIC では，これを次のように書く．

1）ŌPEN文によりファイルを開く

2）FIELD文によりレコードをフィールドに分ける

3）数値データーを文字列型に変換する

4）その結果をバッファーに格納する

5）PUT文によりディスクへの書き込みを行なう

（3）〜5）：出力の場合）

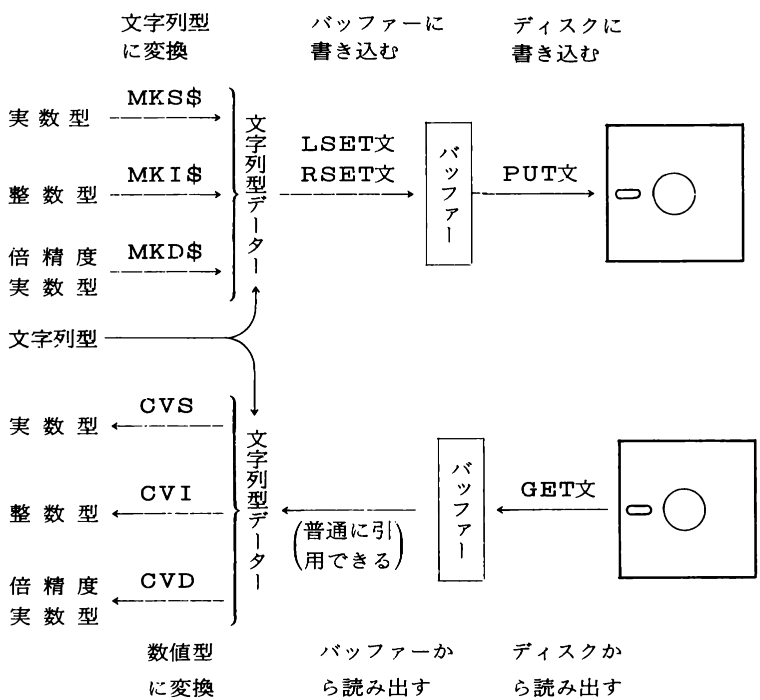
6）GET文によりディスクから1レコード読む

7）フィールド別に文字列データーとしてとり出す

8）それを（必要なら）数値データーに変換する

（6）〜8）：入力の場合）

9）CLŌSEによりファイルを閉じる



ランダム・アクセス・ファイルのオープン

MZ-80 では次のように書く

XŌPEN　#番号，"$FDドライブ番号：ファイル名"

PC-8001，PC-8801 では次のように書く．

ŌPEN　"ドライブ番号：ファイル名"　AS　#番号

Level-3，Micro-8 では次のように書く．

ŌPEN　"R"，#番号，"ドライブ番号：ファイル名"

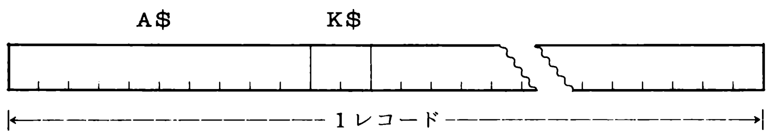
FIELD 文　　これは次の形で書く．

FIELD　#場号，バイト数　AS　フィールド名*

（「バイト数　AS　フィールド名」の部分について）必要ならば，この形をくりかえしてもよい

(例)　FIELD #1,10 AS A$,2 AS K$

これは「1番のファイルに関しては，レコードの最初の10バイトをA$，それに続く2バイトをK$にする」という意味になる．



LSET 文，RSET 文　　これは，

LSET　フィールド名＝文字列式

RSET　フィールド名＝文字列式

の形で書き，それぞれ「左づめにして代入せよ」「右づめにして代入せよ」という意味になる．左辺（代入先のフィールド）のバイト数よりも右辺（代入すべき文字列）のバイト数の方が多い場合，はみ出した部分は無視される．

[注意]　フィールドへの代入は，LSET文またはRSET文に限る．普通の代入文では書き込めない．

---

*　フィールド名の付け方の規則は文字列型変数名と同じである．

**関数 MKS$, MKI$, MKD$**　　LSET文やRSET文でフィールドに代入できるのは文字列型のデーターに限る．数値型のデーターを代入したければ，あらかじめ文字列型に変換しておく必要がある．それには関数STR$を使うのが一案であるが，STR$で変換すると

2進法→10進法の変換に時間がかかる

バイト数が多くなる（たとえば整数型データーは2バイトで表現されているが，STR$で変換すると符号および最大5桁の数字が並ぶから最悪の場合には6バイトになる）

という難点がある．プリンターなどに出力するのであれば，このように，まともに変換する必要があるが，ディスクに入れるのが目的ならば，2進→10進変換などは必要なく，2進法のビット・パターンをそのまま（整数型ならば2バイト，実数型ならば4バイト，倍精度実数型ならば8バイトの文字列として）扱ってくれればよい．そのための形式的な関数が

MKS$（実数型変数名または式）

MKI$（整数型変数名または式）

MKD$（倍精度実数型変数名または式）

である．これらの関数を用いると，ビット・パターンはそのまま変えずに，データーの型を文字列型に変更してくれる．

（例）

```
10 I%=65*256+66
20 C$=MKI$(I%)
30 PRINT C$
```

というプログラムを実行してみるとよい．最初の文を実行すると，整数型変数I%値は16706となり，内部では2進法

0100000101000010

で表されている．2番目の文を実行すると，これが2バイトの文字列

01000001　　01000010

文字Aを表す　　文字Bを表す

とみなされて文字列型変数C$に代入される．最後の文では，それを出力するから，文字列ABが表示される．

関数 CVS, CVI, CVD　これらは前記MKS$，MKI$，MKD$の逆関数で，それぞれ4バイト，2バイト，8バイトの文字列型データーのビット・パターンをそのまま，実数型，整数型，倍精度実数型のデーターとみなして扱う機能をもつ．

PUT 文と GET 文　PUT文はバッファーの内容をディスクに書き込む命令，GET文はディスクの内容をバッファーに読み出す命令である．書き方は次のとおり，

PUT　#番号，レコード番号

GET　#番号，レコード番号

関数 LOC および LOF　LŌCは「最も最近，読み書きしたレコードの番号プラス1」，LŌFは「これまでに書き込んだレコード番号の最大値」を与える．LŌCはレコード番号順に読み書きする場合に便利で，その場合LŌCの値が「次に読み書きすべきレコードの番号」になる．LŌFは新しいレコード番号を割り当てる場合に便利で，その場合，LŌFの値プラス1を新しいレコード番号にすればよい．引数はファイル番号（#の次に書いている番号）である．

(例)　I=LŌC(1)

ファイル#1の最も最近アクセスしたレコードの番号がIに入る．

索　　引

英字・プログラム用語索引

著者略歴

戸川隼人（とがわ はやと）

1935年　東京に生まれる
早稲田中学，早大高等学院を経て
1958年　早稲田大学第一理工学部数学科卒，
科学技術庁航空技術研究所研究員
初年度は東大に出向して真空管式計算機ＴＡＣの開発に従事。以後，数値計算法，計算機科学（図形処理，数式処理，文献検索等），ロケットの軌道計算，自動計測処理システム，構造解析等の研究を行なう。1965年より計算第１研究室長として計算センタ運営にあたる。
1971年　京都産業大学理学部計算機科学科助教授（72年教授）。数値解析学およびシステム工学を担当
1976年　日本大学理工学部数学科教授
数値解析および計算機械論を担当

主要著書

数値計算入門　FORTRANによる有限要素法入門
演習 FORTRAN とその応用　有限要素法概論
数値解析とシミュレーション　共役勾配法
基本 BASIC とその応用

パソコンライブラリ＝2
各種パソコン用
拡張BASICとその応用

昭和57年7月10日©　　初版発行

著者　戸川隼人　　発行者　森平勇三
印刷者　石野昭夫
製本者　徳住恒市

発行所　**株式会社　サイエンス社**
〔本社：営業〕〒101　東京都千代田区神田須田町2-6
TEL　東京（03）256-1091(代)　振替　東京7-2387
〔分室：編集〕〒101　東京都千代田区神田須田町2-4
TEL　東京（03）256-1093(代)

印刷　博文社　　製本　徳住製本所

《検印省略》

3341-80521-2819

基本BASICとその応用

戸川隼人著
A5・1700円

パソコンなど計算機に初めて手を触れようとする人々のために，計算機の基本的知識と，初心者のための言語であるBASICをやさしくしかも体系的に解説．
主要目次　コンピューターの概要（原理，構成，制御方式他）　プログラミング言語　BASIC（くりかえし，操作法他）　プログラミング技法（数値の扱い，データ処理他）　数値計算（近似式，補間，逆行列他）

マイクロ*ソフトBASIC詳説

木下　恂著
近　刊

マイクロソフト系BASICのほぼ完璧な文法書．初心者には独習用入門書となり，熟練者には細かい文法規則を調べるためのレファレンスマニュアルとなる．
主要目次　BASIC入門（基礎知識，操作法，基本要素，入出力，判断と分岐，関数とサブルーチン，乱数，文字の処理他）　拡張仕様（一般事項，判断と分岐，配列，関数とサブルーチン，乱数，文字の処理他）　応用プログラム

サイエンス社

3341-80521-2819

定価1700円

各種パソコン用
拡張BASICとその応用
戸川隼人＝著
パソコンライブラリ＝2
サイエンス社